PC-9801シリーズ テクニカルノウハウ

PC-Techknow

BNN
Bug News Network

Writing
DMSC

98V



PC-TECHKNOW 98V
Hardly a day goes by that technology doesn't touch our lives in some new way. Everything is getting faster, cheaper, and more confusing. And one of the most exciting things happening is the information and telecommunications age that is dawning all around us. ....

PC-9801シリーズ テクニカルノウハウ

PC-Techknow

BNN
Bug News Network

Writing DMSC

98V



PC-TECHKNOW 98V
Hardly a day goes by that technology doesn't touch our lives in some new way. Everything is getting faster, cheaper, and more confusing. And one of the most exciting things happening is the information and telecommunications age that is dawning all around us. ....

発刊にあたって

本書は，V 30 を CPU とする NEC のパーソナルコンピュータ PC 98 V シリーズを対象として，ハードウェアからソフトウェア，開発環境に関する事柄まで，その活用法や開発手法を解説したものです．本書の執筆，ならびに掲載プログラムに関しては，次の点が配慮されています．

● プログラムは，できるだけ従来の PC-9801 シリーズ(PC-9801・PC-9801 E・PC-9801 F・PC-9801 M)でも動作できるように配慮しましたが，PC-98 V シリーズ(PC-9801 U・PC-9801 VF・PC-9801 VM・PC-9801 UV)の独得な機能を使った以下のプログラムは正常な動作が保証できません．

・1-1-3　アナログ RGB テストプログラム
・4-1-2　16 色グラフィックボード対応グラフィックツール

また，使用するオペレーション・システムや開発言語・ライブラリによっては，従来の PC 9800 シリーズで正常な動作が得られない場合もあります．

● 本書に掲載されたプログラムの動作に必要な環境を列記します．

1-1　アナログ RGB テストプログラム
　N 88-DISKBASIC(86)　version 3.0
　アナログ RGB ディスプレイと同ケーブル
1-3　ベンチマークテスト・プログラム
　N 88-DISKBASIC(86)　バージョンは問いません

2-1　システムサブルーチンのテストプログラム，及びベクタアドレス検索プログラム
　N 88-DISKBASIC(86)　バージョンは問いません
2-5　ソフトウェアリセットコマンドの追加プログラム
　N 88-DISKBASIC(86)　version 3.0

3-1　テストプログラム
　MS-DOS version 2.11 又は version 3.10
　MASM(マクロアセンブラ)
　MS-DOS 版 N 88-日本語 BASIC(86)インタープリタ version 3.0
3-2　テストプログラム
　MS-DOS version 2.11 又は version 3.10
　Lattice C コンパイラ version 3.0
　MASM(マクロアセンブラ)
　LINK(MS-LINKER) version 3.0

はじめに

現在，我々の生活空間には様々なコンピュータ機器があります．時代は，既にコンピュータを単なるマニアの所有物から，一般的な道具として認めているようにも思えます．16ビットパーソナルコンピュータ・PC-9801シリーズもワードプロセッサや表計算など実用に耐えうる道具として人気を得て，昨年の1985年にハイ・パフォーマンスな後継機Vシリーズが発表されました．向上した機能にあわせて，高度なアプリケーション・ソフトウェアも次々と発売され，プログラミングというパーソナルコンピュータを使用する上で最も創造的と思われる作業も，次第に一般利用者に縁遠い存在となってきました．

エキスパートシステム，AI，ワークステーション，周辺機器とのインターフェース，利用環境の改善…など，高度なソフトウェアの開発手法に関して詳しい説明をしている書物があまりにも少なく，コンピュータの技術の進歩に一般利用者がついていくことは難しい状況になっています．

本書では，このような状況のもとに、高度なアプリケーション・ソフトウェア開発のノウハウをできるだけ詳細に記述するように心がけ，読者のコンピュータの技術進歩に対する関心に答えようとしています．また，PC-9800シリーズの内部的な解析に伴って知り得た情報も詳細に記述することにより，より優れたアプリケーション・ソフトウェアが開発されるよう考慮しました．

いまや，プログラミングもシステマティックな開発環境を必要とする時代です．大きく世代交代をしようとしているパーソナルコンピュータの世界．本書が，そこで活躍しようとしている皆さんのお役に立てれば望外に幸いです．

安井　勉

目次

PC-Techknow 98V

第2章　BASIC ROMの解析／新コマンド追加法　19

3-3 TURBO PASCAL version 3.0 191

3-4 MS-DOSマクロアセンブラ 198

第4章 拡張インターフェースボード, 周辺機器の利用 221

4-1 16色グラフィックボード対応タイニーグラフィックツール 221

付　録

PC-9801シリーズの特徴

第1章

1-1　従来機とVシリーズ

1-1-1　カタログ上での相違点

NECのPC-9801シリーズは1982年10月に発売されて以来，すでに3年が経過しています．発売当時のパーソナルコンピュータといえば8ビット機が主流であり，16ビットCPU(i 8086)を使用したPC-9801シリーズはビジネスユースにも十分なスペックをもっていました．

その後，PC-9801シリーズも技術の進歩や時代の要求にあわせて機能を強化していきマイナーチェンジを繰り返しました．この項では，現在のPC-9801 Vシリーズの特徴を知るためにPC-9801シリーズの歴史を振り返ってみます．

1-1-1-1　PC-9801

PC-9801は，PC-8801の上位機種としての位置づけをもって発表されました．PC-8801シリーズとの大きな違いはキーボードで，ファンクションキーやBSキー・XFERキーの追加，カーソルキーの独立などの変更が加えられ，漢字を扱えるパーソナルコンピュータとしての配慮がなされています．キーの数が増えた分，キーボードのサイズは480(W)×210(D)×63(H)mmとPC-8801シリーズと比較して大きいものとなりました．本体裏面の拡張スロットは6つで，8インチ両面倍密度フロッピーディスクインターフェイスを標準装備しています．

また，PC-8801シリーズとBASICレベルでの互換性がほぼ保証されており，機能や動作速度の点でPC-8801シリーズのそれを大きく上回っています．まず，グラフィックV-RAMが2倍の96Kバイトになったため，640×400ドットでカラー8色の表示が可能となりました．グラフィックディスプレイコントローラ(μPD 7220)の採用によりグラフィック命令の実行速度も飛躍的に向上しています．

演算コ・プロセッサ・i 8087(別売りオプション)を使用することによって，三角関数や平方根の計算速度は5倍から10倍の向上が期待できます．

1-1-1-2　PC-9801 E

PC-9801 EはPC-9801の周辺チップをカスタムIC化し，本体の小型化とコストダウンを目指したものです．

外形寸法は，本体・キーボードともにPC-9801より小型になり，重量も軽くなっています．最

も目につく改良点はキーボードにあります．ケーブルを細くすることによって，キーボードの取り回しがとても楽になり，コネクタを本体裏面に移すことによって，誤って抜いてしまうトラブルを防いでいます．また，キーボード形式もシリンドリカル型からスカルプチャ型に変更しています．

本体正面にはクロック切り換えスイッチが追加され，PC-9801では5MHz固定だったクロック周波数を8MHzに切り換えることによって，実行速度を速くすることもできます．本体裏面からは8インチフロッピーディスクインターフェイスがなくなりました．

PC-9801 EはPC-9801とソフトウェアコンパチビリティを保っていますが，内部は大幅に変更されています．CPUには8MHzタイプのi8086-2が使われています．周辺チップのカスタム化も進み，メインボード基盤そのものも随分小型となりました．グラフィックV-RAMはPC-9801の2倍である192Kバイトになりました．これにより，8色の640×400ドットを2画面，モノクロで640×200ドットならば12画面使用することができます．

1-1-1-3　PC-9801 F

PC-9801 FはPC-9801 Eに640Kバイトタイプの両面倍密度倍トラックフロッピーディスクドライブを本体に搭載したものです．ディスクドライブ1台内蔵のものをPC-9801 F1，2台内蔵したものをPC-9801 F2，1台と10Mバイトのハードディスクを搭載したものをPC-9801 F3と呼びます．ディスクドライブが本体に実装されたことにより，幅・奥行きはPC-9801 Eと同じ寸法ですが，高さはPC-9801 Fの方が25mm高くなっています．本体裏面の拡張スロットは4つに減り，増設用の5インチ両面倍密度倍トラックフロッピーディスクインターフェースが追加されました．また，ディスクドライブなどの熱による障害を防止するため，クーリングファンも装備されています．

PC-9801 Fはフロッピーディスクドライブの実装を除くと，ほとんどPC-9801 Eと同一のマシンと言えます．それまで別売りオプションであったJIS第一水準漢字ROMが標準装備となり，これにともなってN88-日本語BASIC(86)が付属しています．かな漢字変換入力ができるようになったため，より日本語処理を身近なものとしています．

1-1-1-4　PC-9801 M

PC-9801 Mは，PC-9801 Fの本体に実装されている640Kバイトタイプのフロッピーディスクドライブを1Mバイトタイプの両面高密度フロッピーディスクドライブに変更し，標準実装メモリを増設したものです．ディスクドライブ2台内蔵のものをPC-9801 M2，1台と20Mバイトのハードディスクを搭載したものをPC-9801 M3と呼んでいます．外形寸法などの，目につく変更はほとんど無く，フロッピーディスクドライブのアクセスランプがPC-9801 Fのそれよりも小さくなったことぐらいです．

1Mバイトタイプ両面高密度フロッピーディスクの構造は8インチ両面倍密度フロッピーディ

スクとまったく同じです．従って，増設用両面高密度フロッピーインターフェースに，外付用の8インチ両面倍密度フロッピーディスクドライブを接続することも可能です．

1-1-1-5 PC-9801 U

PC-9801 Uから，PC-9800シリーズの機能は大きく変更されました．まず，CPUがi 8086から，NEC自社開発のV 30(μPD 70116 8 MHz)に変更されました．V 30を使用することによって，i 8086を使用していた従来機との上位互換性を保ちながら実行速度を向上させています．

グラフィック機能も強化され，アナログRGBカラーディスプレイを接続することによって，4096色中の8色表示(オプションの16色グラフィックボードの装着すれば，16色)が可能となりました．

また，本体に実装されるフロッピーディスクドライブを640 Kバイトタイプの3.5インチ両面倍密度倍トラックフロッピーディスクドライブに変更することによって，かなりコンパクトなサイズになりました．加えて，ディップスイッチを本体正面に移動し，ボリューム調整つまみを追加しました．本体裏面にはマウスコネクタとジョイスティックコネクタが新設されました．拡張スロットの数は2つに減っていますが，増設RAMボード，16色グラフィックボード，サウンドボードはそれぞれ専用スロットを用意しています．キーボードはキートップに丸みを付けることによってミスタイプを少なくし，全体に小型になりました．JIS第1水準漢字ROM，JIS第2水準漢字ROMも標準装備，N 88-日本語BASIC(86) version 3.0(文節変換による漢字入力が可能)など日本語処理機能は，さらに強化されています．また，オプションのFMシンセサイザーを装着することにより，FM音源3音+PSG音源3音の合わせて6重和音による音楽演奏が可能になっています．グラフィックV-RAMは，従来機と比べると半分の96 Kバイトとなり，640×400ドットカラー8色の画面を1画面しか持つことができません．

1-1-1-6 PC-9801 VF

PC-9801 VFはPC-9801 Fの上位バージョンとして発表されました．640 Kバイトタイプの両面倍密度倍トラックフロッピーディスクドライブを2台搭載したものです．また，PC-9801 Uと同様に，CPUがV 30(μPD 70116 8 MHz)に変更されました．

外形のデザインがPC-9801 Uと同時期に発売されたPC-98 XAと似たスタイルとなり，フロッピーディスクドライブは右端に寄せられた形になりました．サイズはPC-9801 F／Mと全く同じです．

PC-9801 Uと同じく，キートップに丸みが付き，JIS第1水準漢字ROM，JIS第2水準漢字ROM，またN 88-日本語BASIC(86) version 3.0の採用など，使いやすさに配慮がなされています．尚，グラフィックV-RAMは，PC-9801 E／F／Mと同じように192 Kバイトもっていますので，PC-9801 Uとは違い640×400ドットカラー8色を2画面持つことができます．また，メインメモリは256 Kバイトを標準実装しています．

1-1-1-7　PC-9801 VM

PC-9801 VMはPC-9801 Mの上位バージョンとして発表されました．本体に実装されているフロッピーディスクドライブは，5インチ2DD／2HDの両方のタイプのフロッピーディスクをメディアの違いを意識することなく使用できるものです．このタイプのフロッピーディスクドライブを2台搭載したモデルをPC-9801 VM 2と呼び，本体には搭載されていませんがフロッピーディスクのインターフェースを持つモデルをPC-9801 VM 0と呼びます．PC-9801 VM 4はフロッピーディスクドライブ2台に加え，20 Mバイトのハードディスクを搭載しています．CPUはV 30(μPD 70116-10　8／10 MHz)に変更され，クロック周波数は，8 MHz／10 MHzの切り替えが可能です．

PC-9801 VM 0とPC-9801 VM 2の外形のデザインは，PC-9801 VFとクロックの切り替えスイッチを除けば，まったく同じものです．PC-9801 VM 4はそれより若干大きいサイズです．メインメモリはさらに大きくなり384 Kバイトを実装しています．

1-1-1-8　PC-9801 UV

PC-9801 UV 2はPC-9801 Uの上位バージョンとして発表されました．1 Mバイトタイプの両面高密度フロッピーディスク，640 Kバイトタイプの両面倍密度倍トラックフロッピーディスクの両方に使える3.5インチディスクドライブを2台搭載したものです．ボディのサイズはPC-9801 Uと全く同じコンパクト・サイズです．キーボードコネクタは，本体正面に戻りました．

PC-9801 UVでは，いままでオプションであったサウンドボード，16色グラフィックボードが標準装備となりました．グラフィックV-RAMも標準で256 Kバイトもっているため，アナログRGBディスプレイの使用により4096色中16色を使うことができます．メインメモリはPC-9801 VMと同様に384 Kバイト実装です．



PC-9801 VM 2



PC-9801 UV 2

1-1-1-9　PC-9801 VM 21

PC-9801 VM-21は従来のPC-9801 VMシリーズの後継機種としての位置づけがされており，カタログ上のスペックにおいては上位機種であるといえます．

本体背面のインターフェース類の位置が若干変更されており，キーボードのコネクタが前面パネルに移動したことが最も目につく変更点であります．ボディ，キーボードともモデファイを受けており，いくらか丸みを帯びたデザインに変更されていますが，キーの位置や種類などの大きな変更点はありません．

本体実装メモリーを640Kバイトに増設し，また，アナログ4096色中16色を同時使用できるなどのグラフィック機能面での性能の強化などが行われ，実務に利用する大量のデータもスピーディに扱えるようになりました．当然のことですが，PC-9801 VMシリーズのソフトウェアは完全な互換性をもって動作させることができます．



PC-9801 VM 21

1-1-1-10　PC-9801 VX

PC-9801 VXは従来のPC-9801 VMシリーズの上位機種で，速度の向上と実装メモリーの拡張を目的としてインテル社製i 80286 CPUをμPD 70116-10と共に実装しています．i 80286モードとμPD 70116-10の切り替えはディップスイッチによって行われ，i 80286モードにおいてもPC-9801 VMシリーズの一部のソフトウェアを動作させることが出来るのが特徴です．また，i 80286モードでの動作スピードは，μPD-70116-10の場合に比べて1.3から1.7倍に向上するようです．

本体は，PC-9801 VM 21と同じく丸みを帯びたデザインにモデファイを受けており，本体背面のインターフェース類の位置も同じように変更を受けています．

実装メモリーは640Kバイトを標準実装していますが，i 80286を使用する場合には最大4.6Mバイトまで増設することが可能です．また，グラフィック機能面では，アナログRGB 4096色中／16色表示ができ新規開発のEGCのため，処理スピードを大幅に向上させています．

なお,PC-9801 VX に関しては VX 0／VX 2／VX 4 の 3 機種があり,PC-9801 VM シリーズと同様,内蔵のディスクドライブの数やタイプによって分類されています.

使用できるソフトウェアですが,μPD 70116-10 モードでは PC-9801 VM シリーズと完全な互換性をもっており,すべてのソフトウェア・周辺機器を使用することができます.また,i 80286 モードにおいても一部のプログラムをそのまま動作させることが可能ですので,速度の点から複雑で大きなアプリケーションも動作させることが出来るようになりました.



PC-9801 VX

機種名	PC-9801	PC-9801E	PC-9801F2	PC-9801F3	PC-9801M2	PC-9801M3
サイズ　本体 （mm）キーボード	500×400×125 480×210×63	420×345×125 470×195×38	420×345×150 470×195×38	420×345×150 470×195×38	420×345×150 470×195×38	420×345×150 470×195×38
重　量　本体 （kg）キーボード	9.6 1.6	7.5 1.6	9.6 1.6	10.9 1.6	10.0 1.6	10.9 1.6
CPU	i8086	i8086-2	i8086-2	i8086-2	i8086-2	i8086-2
ROM	96KB	96KB	96KB	96KB	96KB	96KB
RAM	128KB	128KB	128KB	256KB	256KB	256KB
VRAM　テキスト グラフィック	12KB 96KB	12KB 192KB	12KB 192KB	12KB 192KB	12KB 192KB	12KB 192KB
漢字 ROM	なし	なし	第1水準	第1水準	第1水準	なし
フロッピーディスク	なし	なし	5'2DD×2	5'2DD×1 10MB・HD	5'2HD×2	5'2HD×1 20MB・HD
インターフェイス	プリンタ RS-232C 320KB・FD 1MB・FD	プリンタ RS-232C 320KB・FD	プリンタ RS-232C 320KB・FD 640KB・FD	プリンタ RS-232C 320KB・FD 640KB・FD	プリンタ RS-232C 320KB・FD 1MB・HD	プリンタ RS-232C 320KB・FD 1MB・FD 20MB・HD
スロット	4	6	4	4	3	3
価格	298,000	215,000	398,000	758,000	415,000	830,000

機種名	PC-9801U2	PC-9801VF2	PC-9801VM0	PC-9801VM2	PC-9801VM4	PC-9801UV2
サイズ　本体 (mm)　キーボード	398×335×87 435×180×34	420×345×150 470×195×38	420×345×150 470×195×38	420×345×150 470×195×38	470×345×150 470×195×38	398×335×87 435×180×34
重　量　本体 (kg)　キーボード	5.6 1.2	10.3 1.6	8.3 1.6	10.3 1.6	12.5 1.6	5.6 1.2
CPU	μPD70116-8	μPD70116-8	μPD70116-10	μPD70116-10	μPD70116-10	μPD70116-10
ROM	96KB	96KB	96KB	96KB	96KB	96KB
RAM	128KB	256KB	384KB	384KB	384KB	384KB
VRAM　テキスト グラフィック	12KB 96KB	12KB 192KB	12KB 192KB	12KB 192KB	12KB 192KB	12KB 256KB
漢字 ROM	第1・2水準	第1・2水準	第1・2水準	第1・2水準	第1・2水準	第1・2水準
フロッピーディスク	3.5'2DD×2~	5'2DD×2	なし	5'2DD／2HD×2	5'2DD／2HD×2 20MB・HD	3.5'2DD ／2HD×2
インターフェイス	プリンタ RS-232C マウス 640KB・FD	プリンタ RS-232C マウス 640KB・FD	プリンタ RS-232C マウス 1MB・FD	プリンタ RS-232C マウス 1MB・FD	プリンタ RS-232C マウス 1MB・FD	プリンタ RS-232C マウス 1MB・FD
スロット	2	4	4	4	4	2
価格	298,000	348,000	298,000	415,000	830,000	318,000

機種名	PC-9801 VM 21	PC-9801 VX 0	PC-9801 VX 2	PC-9801 VX 4
サイズ　本体 (mm)　キーボード	420×345×150 470×195×38	420×345×150 470×195×38	420×345×150 470×195×38	470×345×150 470×195×38
CPU	μPD 70116-10	i 80286 相当品 μPD 70116-10	i 80286 相当品 μPD 70116-10	i 80286 相当品 μPD 70116-10
ROM	96 KB	96 KB	96 KB	96 KB
RAM	640 KB	640 KB	640 KB	640 KB
VRAM　テキスト グラフィック	12 KB 256 KB	12 KB 256 KB	12 KB 256 KB	12 KB 256 KB
漢字 ROM	第1・2水準	第1・2水準	第1・2水準	第1・2水準
フロッピーディスク	5'2 DD／2 HD×2	なし	5'2 DD／2 HD×2	5'2 DD／2 HD×2 20 MB-HD
インターフェイス	プリンタ RS-232 C マウス 1 MB-FD	プリンタ RS-232 C マウス 1 MB-FD	プリンタ RS-232 C マウス 1 MB-FD	プリンタ RS-232 C マウス 1 MB-FD 20 MB-HD
スロット	4	4	スロット	4
価格	390,000	353,000	433,000	693,000

表 1-1-1　PC-9801 シリーズ仕様比較表

1-1-2 アナログRGBテストプログラム

PC-9801 U／VF／VM シリーズでは，N 88-日本語 DISKBASIC version 3.0 を使用することによって，4096色中8色まで表示することができます．BASICから利用するには，COLOR文の第5パラメーターを1にして，アナログRGB 4096色中8色表示モードに設定します．

次に例を示します．

color ,,,,0　　デジタルRGB(8色モード)
color ,,,,1　　アナログRGB(4096色中8色モード)
color ,,,,2　　アナログRGB(4096色中16色モード)

プログラム1-1-1は，640×400ドットの画面を8分割して0～8までのパレットで四角く塗りつぶした後，color=(,)文によってパレットに0から4096色(&H 0～&HFFF)までのカラーコードを順番に入れている簡単なものです．スペースキーを押すことによって，画面上の四角の色が変化します．

次に例を示します．

color=(0,&HFFF)　　パレット0に,&HFFFのカラーコードをセット

プログラム1-1-1　アナログRGBテストプログラム

```
100 '*************************************************************************
110 ' アナログＲＧＢ テストプログラム By [Benny] copyright(C) DMSC  1986.5.26
120 '*************************************************************************
130 '
140  CONSOLE 0,25,0,1 : SCREEN 3,0,0,1
150  COLOR ,,,,1       : CLS 3
160 '
170  LINE(  0,  0)-(160,200),0,BF : LINE(160,  0)-(320,200),1,BF
180  LINE(320,  0)-(480,200),2,BF : LINE(480,  0)-(640,200),3,BF
190  LINE(  0,200)-(160,400),4,BF : LINE(160,200)-(320,400),5,BF
200  LINE(320,200)-(180,400),6,BF : LINE(480,200)-(640,400),7,BF
210 '
220  FOR IRO=0 TO &HFFF STEP 8
230   PALETTE=0
240   FOR RT=IRO TO IRO+7
250   A$=INKEY$ : IF A$="" THEN 250 ELSE LOCATE 0,PALETTE*2
260          COLOR=(PALETTE,RT) : PALETTE=PALETTE+1
270          PRINT "PALETTE(";PALETTE;")=";RT
280          PRINT
290   NEXT RT
300  NEXT IRO
310 '
320 END
```

1-2 メモリマップ

普通，パソコンをアプリケーションやゲームだけに使うのなら，メモリマップを意識することはありません．また，プログラムを組む場合でも，BASICだけで行うのならば，コンピュータ内部のメモリ構成がどうなっているのか知らなくてもプログラミングはできます．

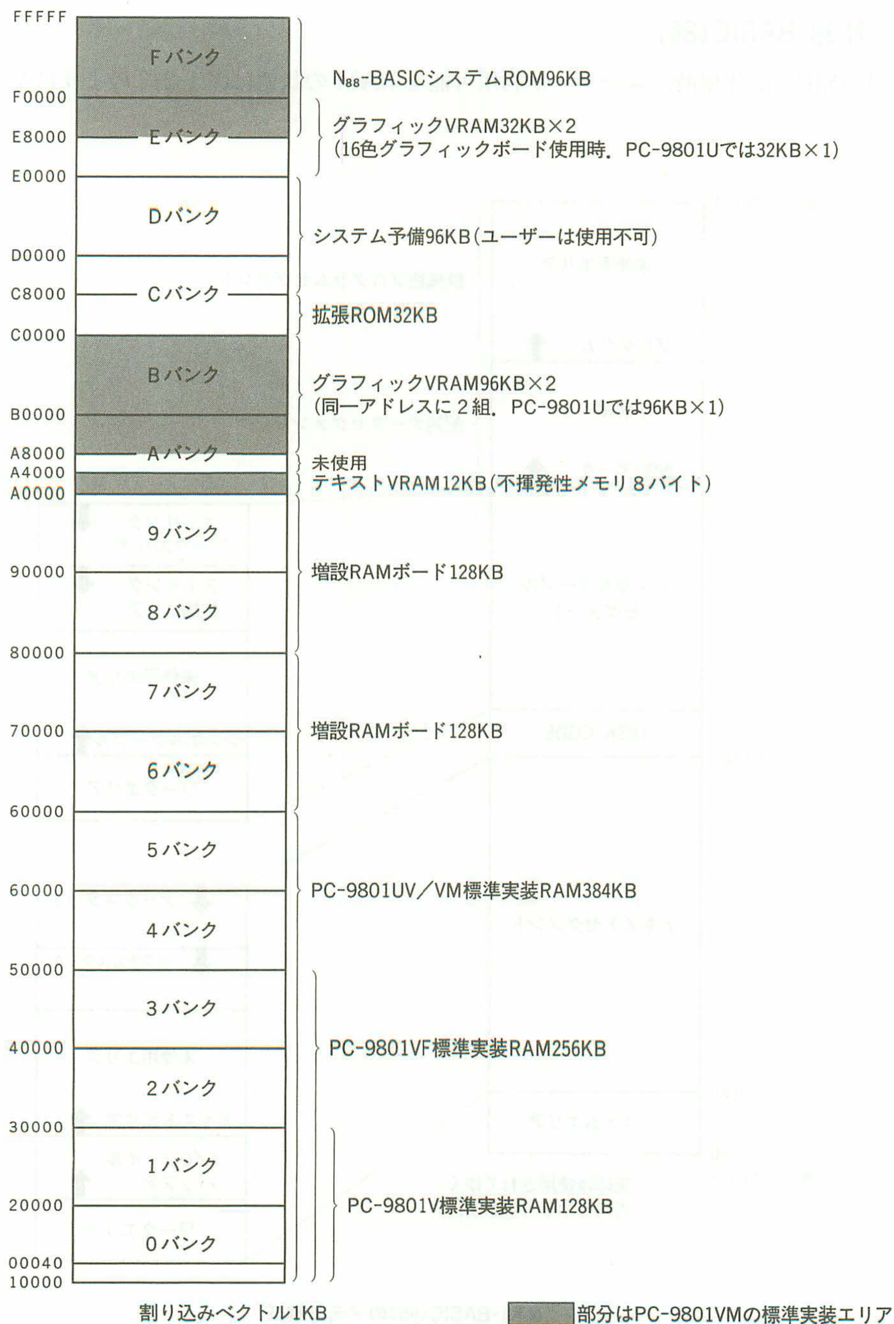


図1-2-1　PC-9801 U／VF／VM全アドレスのメモリ構成

しかし，効率の良さ，実行速度の速さが要求されるプログラムでは，機械語の使用が不可欠になってきます．その場合，コンピュータ内部のメモリ構成を知らなくてはプログラミングはできません．また，BASICやコンパイル型の高級言語の環境下で機械語サブルーチンを使用する場合も同様です．

PC-9801 U／VF／VMの全メモリ・エリアの概要を図1-2-1に示します．

1-2-1　N 88-BASIC(86)

N 88-BASIC(86)使用時，ユーザーが利用可能なRAMの状態は図1-2-2のようになっています．

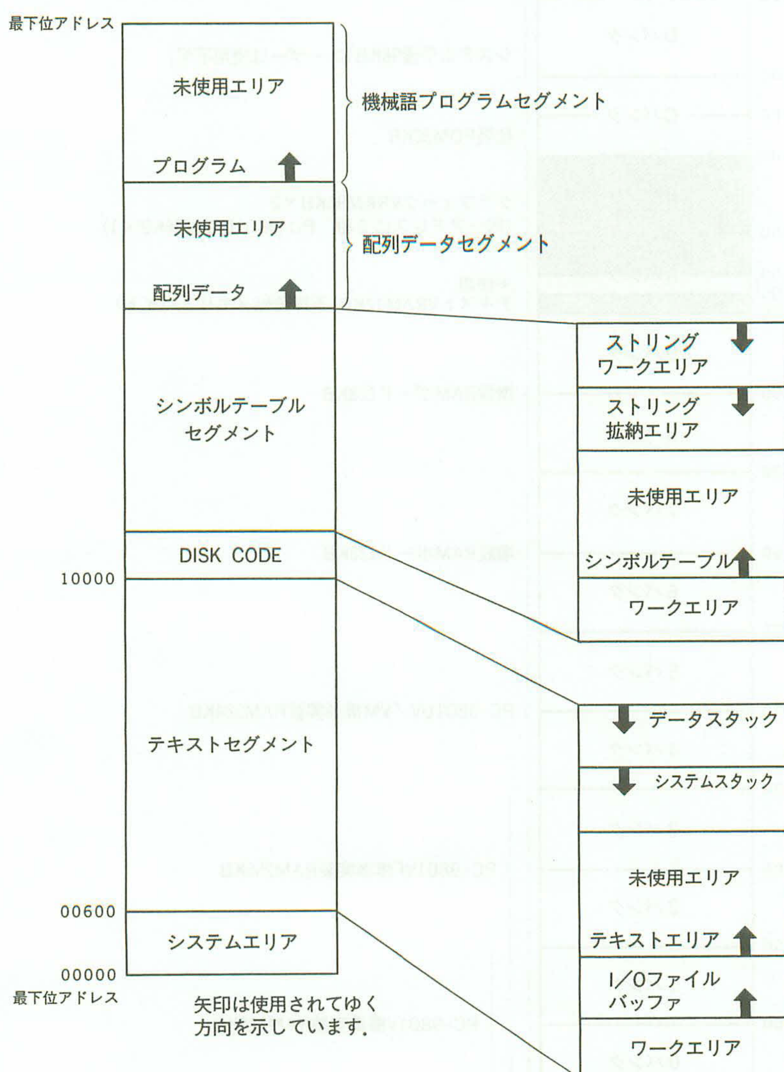


図1-2-2　N 88-BASIC(86)のメモリ構成

RAM 全体はシステム・エリアと 4 つのセグメントに分けられています．下位アドレスから順に解説していきます．

a.システム・エリア　[物理アドレス 0 H～00600 H]

この領域はシステム・エリアとして，割り込みベクタ，入出力制御作業エリア等に使われます．

b.テキスト・セグメント　[物理アドレス 00600 H～10000 H]

オフセットアドレス 0 H～1 AFFH は BASIC インタプリタのワークエリアとして使われます．

I／O ファイル・バッファ領域の大きさは，システム立ち上げ時に設定される同時オープン可能ファイル数によって可変です．

テキスト・エリアには BASIC プログラムが下位アドレスから上位アドレスの方向に格納されていきます．なお，未使用テキスト・エリアの大きさは FRE(1) 関数によって知ることができます．

システム・スタック・エリアは上位アドレスから下位アドレスの方向に使用されていきます．

データ・スタック・エリア(演算スタック)は上位アドレスから下位アドレスの方向に使用されていきます．なお，CLEAR 命令の第 3 パラメータによって大きさを変えることができます．オプションの拡張機能を使用する場合，テキスト・エリアの一部をシステムが使用するので，その分テキスト・エリアが少なくなります．

c.DISK CODE　[物理アドレス 10000 H～*****]

DISK BASIC モード時，この領域に DISK CODE 部が読み込まれます．

DISK　CODE 部の大きさはシステムの起動形態によって可変ですが，標準 DISK　CODE の大きさは 24 KB です．

システムの起動形態と標準 DISK CODE に加算されるメモリサイズの関係は次の通りです．

文節変換入力方式日本語入力機能	42 KB
表示選択入力方式日本語入力機能	14 KB
拡張グラフィックモード	16 KB
拡張画面ハードコピー機能	6 KB
モニタ・モードのディスク機能	11 KB

d.シンボルテーブル・セグメント

シンボルテーブル・セグメント全体の大きさは配列データ・セグメントの大きさの増減によって可変です．

シンボルテーブル・エリアにはラベル，変数，関数名と属性，数値型変数データの値，文字型変数のストリング・ディスクリプタが下位アドレスから上位アドレスの方向に格納されていきます．なお，未使用シンボルテーブル・エリアの大きさは FRE(0) 関数によって知ることができます．

ストリング格納エリアには文字型変数，配列に代入される文字列が上位アドレスから下位アドレスの方向に格納されていきます．

ストリング・ワーク・エリアは2KBの領域で，文字列の演算作業に使われます．

e.配列データセグメント

配列データ・セグメントの大きさはCLEAR命令の第4パラメータによって変えることができます．

数値型の配列データと文字型配列のストリング・ディスクリプタが下位アドレスから上位アドレスの方向に格納されていきます．なお，未使用エリアの大きさはFRE(3)関数によって知ることができます．

f.機械語プログラム・セグメント

この領域は機械語プログラムを展開するための領域で，BASICシステムに干渉されません．また，機械語プログラム・セグメントの大きさはCLEAR命令の第2パラメータによって変わります．

機械語プログラムは下位アドレスから上位アドレスの方向に展開されていきます．

1-2-2　MS-DOS

MS-DOS立ち上げ時の初期化動作は，ブート・プログラムの実行からCOMMAND.COMをメモリ上に読み込むまでかなり複雑な処理が行われますが，最終的(コマンド待ち状態)には図1-2-3のようなメモリ構成になります．

ただし，図1-2-3に示したメモリマップはCONFIG.SYSファイルによるディスクバッファ数やデバイス・ドライバの指定がない場合のものです．

CONFIG.SYSファイルによってデバイス・ドライバの指定を行っている場合，それらのデバイスドライバはMSDOS.SYSの上位にロードされます．

また，ディスクバッファ数(一度にオープンできるファイル数)の指定を行った場合もそれらのバッファ領域はMSDOS.SYSの上位にロードされます．

1-3　i8086とV30の比較ベンチマークテスト

従来のPC-9801シリーズは，CPUにインテルのi8086(正式名称はiAPX 86)が使われていました．しかし，Vシリーズでは，日本電気が新しく開発したV30(μPD 70116)と呼ばれるCPUが使われるようになりました．ここでは従来のPC-9801シリーズのCPUをV30に差し替えることによって，どの程度，性能が向上するかという実験を行いました．

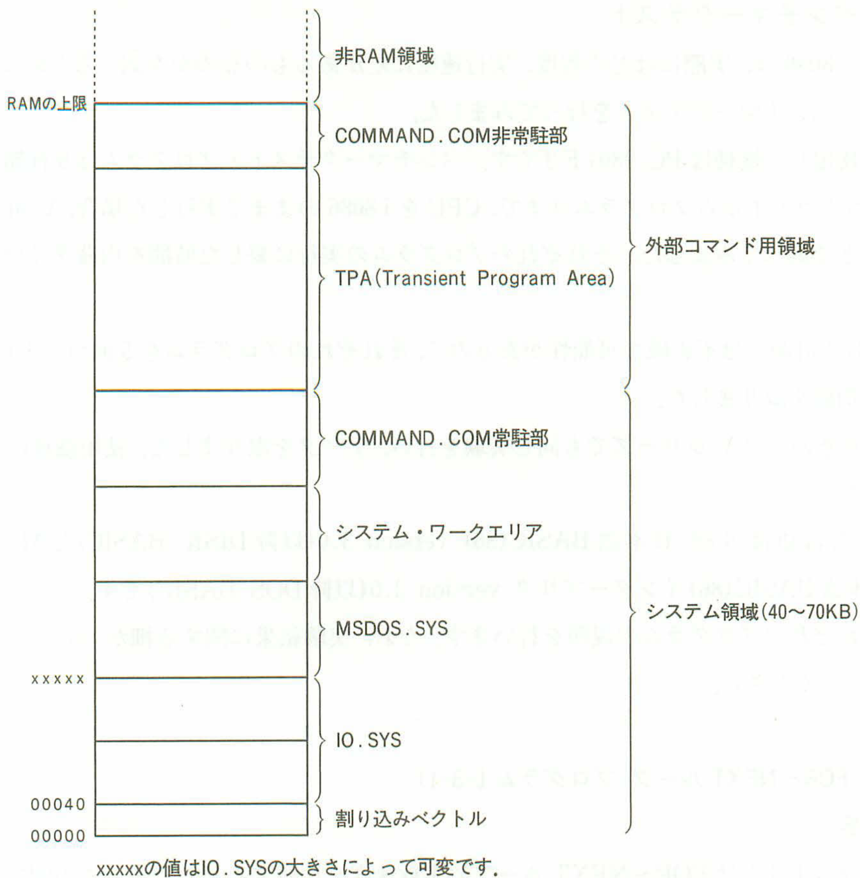


図1-2-3　MS-DOSのメモリ構成

1-3-1　V30の特徴

V30の最大の特徴は，i8086に対して完全な上位互換性を持っていることです．このことによって，V30を搭載したマシンでは，PC-9800シリーズの膨大なソフトウェア資産を無駄にすることなく，さらに速い実行速度で利用することが可能になっています．

V30は様々な機能が新たに付加されていながら，プログラムの実行速度がi8086に比べて格段に速くなっています．これは，i8086の内部でマイクロプログラムによって実行されていた部分を一部，電子回路に置き換えたことによります．特に速くなった部分として，乗算・除算や文字列操作が挙げられます．

また，V30には8080エミュレーション・モードという機能が付加されました．このモードでは，そのまま8080のプログラムを実行することが可能であり，CP/Mを走らせることもそれほど難しいことではありません．

1-3-2　ベンチマークテスト

V 30とi 8086は，実際にはどの程度，実行速度に差があるものなのかを調べるために，次のような方法でベンチマークテストを行ってみました．

実験に使用した機種はPC-9801 F 2です．ベンチマークテスト・プログラムは9種類用意しました．プログラム1からプログラム9まで，CPUをi 8086のままで実行した場合，V 30に差し替えた場合とで調べてみました．それぞれのプログラムの実行に要した時間を内蔵タイマーで計りました．

1回だけの計測では不正確な可能性があるので，それぞれのプログラムを5回づつ実行し，実行時間の平均値を取りました．

また，参考のためVシリーズでも同じ実験を行い，データを取りました．使用機種はPC-9801 VM 2です．

使用した言語はN 88-日本語BASIC(86) version 3.0(以降DISK-BASIC)とMS-DOS版N 88-日本語BASIC(86)インタープリタ version 3.0(以降DOS-BASIC)です．

次にそれぞれのプログラムの説明を行います．なお，実験結果に関する細かいデータは表1-3-1を参照してください．

1-3-2-1　FOR～NEXTループ(プログラム1-3-1)

●実行内容

プログラム1-3-1はFOR～NEXTループの実行速度を比較するものです．この例では6万回のループを行っています．

●結果

DISK-BASIC，DOS-BASIC共に10％前後，実行速度が速くなっています．PC-9801 VMではさらに20％速くなっています．

1-3-2-2　四則演算(プログラム1-3-2，プログラム1-3-7～8)

●実行内容

プログラム1-3-2は整数で四則演算を行います．加算・減算・乗算・除算を1万回実行させてみました．計算内容は，1を足す，1を引く，3を掛ける，3で割る，の4つです．プログラム1-3-7～8はそれぞれ倍精度実数，単精度実数で同じ計算を行いました．

●結果

全てのプログラムでDISK-BASIC，DOS-BASIC共に10％前後，速くなっています．PC-9801 VMでは，さらに20％弱速くなっています．

1-3-2-3　三角関数(プログラム 1-3-3)

●実行内容

プログラム 1-3-3 は倍精度実数で三角関数の計算を行います．三角関数の計算自体に時間がかかってしまうため，ループの回数は 500 回にしました．

●結果

DISK-BASIC では約 8 %，DOS-BASIC では約 27 %程，速くなりました．DOS-BASIC は，このプログラムを DISK-BASIC の 6 分の 1 の時間で実行しました．また，PC-9801 VM では DISK-BASIC の 9 分の 1 の時間で終わっています．

1-3-2-4　LINE 文(プログラム 1-3-4)

●実行内容

プログラム 1-3-4 はグラフィック画面上に線を引きます．このプログラムでは，だいたい 2500 本程引いています．

●結果

DISK-BASIC，DOS-BASIC 共に 5 %速くなっています．他のテストに比べると，あまり劇的な差はありませんが，PC-9801 VM では 40 %も速くなっています．

1-3-2-5　GET@文／PUT@文(プログラム 1-3-5)

●実行内容

プログラム 1-3-5 は GET@・PUT@の速度比較です．グラフィック画面を 64×40 のサイズで取り込み，適当な部分へ移します．5000 回ループしています．

●結果

DISK-BASIC で 4 %，DOS-BASIC で 7 %速くなっています．プログラム 1-3-4 と同様，他のベンチマークテストに比べて，あまり速くなりませんでした．

1-3-2-6　PAINT 文(プログラム 1-3-6)

●実行内容

プログラム 1-3-6 は PAINT のテストです．画面を 40×25 に区切り，順番に色を塗ります．

●結果

DISK-BASIC，DOS-BASIC 共に 10 %前後，速くなっています．PC-9801 VM では 6 割近くも速くなっています．

1-3-2-7　PEEK／POKE 文によるブロック転送(プログラム 1-3-9)

●実行内容

プログラム 1-3-9 では PEEK・POKE 文によるブロック転送を行っています．実行の様子，結

果が見えるようにV-RAMの内容を転送させています．そのため普通のRAMで同じプログラムを走らせるよりも少々時間がかかっているかもしれません．

●結果

DISK-BASIC, DOS-BASIC共に10％前後，速くなっています．PC-9801 VMでは20％弱速くなっています．

1-3-3 考察

今回のベンチマークテストでは，全ての場合に実行速度は向上しました．しかし，実際にはi8086とV30の実行サイクルが違います．i8086では，CPUの稼動時間と待機時間の割合は1：1ですが，V30では2：1になっています．また，PC-9801 F2はもとより，Vシリーズ以前の機種では周辺チップもi8086に合わせて組まれているため，CPUの差替えだけでは，さほど速くならないようです．

プログラム1-3-1 FOR～NEXTループ ①

```
10 REM FOR ～ NEXT ループ
100 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
110 TM=FNT
120 FOR I=-30000 TO 30000
130 NEXT I
140 PRINT FNT-TM
150 END
```

プログラム1-3-2 四則演算(整数) ②

```
10 REM 四則演算（整数）
100 DEFINT A-Z
110 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
120 TM=FNT
130 FOR I=1 TO 10000
140   A=A+1
150   B=B-1
160   C=C*3
170   C=C/3
180 NEXT I
190 PRINT FNT-TM
200 END
```

プログラム1-3-3 三角関数 ③

```
10 REM 三角関数（倍精度）
100 DEFDBL A-D
110 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
120 TM=FNT
130 FOR I=1 TO 500
140   D=I
150   A=SIN(D)
160   B=COS(D)
170   C=TAN(D)
180 NEXT I
190 PRINT FNT-TM
200 END
```

プログラム 1-3-4 LINE 文 ④

```
10 REM グラフィック（LINE）
100 SCREEN 3:CLS 3
110 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
120 TM=FNT
130 FOR I=0 TO 2500
140   LINE (I*.256,399)-(639,399-I*.16),I MOD 7 +1
145   LINE (I*.256,0)-(639,I*.16),I MOD 7 +1
150 NEXT I
160 PRINT FNT-TM
170 END
```

プログラム 1-3-5 GET@文／PUT@文 ⑤

```
10 REM グラフィック（GET@、PUT@）
100 SCREEN 3:CLS 3
110 DIM A%(770)
120 FOR I=0 TO 100
130   LINE(I*6.39,0)-STEP(0,399),I MOD 7 +1
140   LINE(0,I*3.99)-STEP(639,0),I MOD 7 +1
150 NEXT I
160 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
170 TM=FNT
180 FOR I=1 TO 1000
190   GET@(INT(RND*10)*64,INT(RND*10)*40)-STEP(63,39),A%
200   PUT@(INT(RND*10)*64,INT(RND*10)*40),A%,XOR
210 NEXT I
220 PRINT FNT-TM
230 END
```

プログラム 1-3-6 PAINT 文 ⑥

```
10 REM グラフィック（PAINT）
100 SCREEN 3:CLS 3
110 DIM A%(770)
120 FOR I=0 TO 40
130   LINE(I*15.975,0)-STEP(0,399),7
140   LINE(0,I*15.96)-STEP(639,0),7
150 NEXT I
160 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
170 TM=FNT
180 FOR I=0 TO 999
190   PAINT((I MOD 40)*15.975+2,(I¥40)*15.96+2),I MOD 6 + 1,7
210 NEXT I
220 PRINT FNT-TM
230 END
```

プログラム 1-3-7 四則演算（倍精度）⑦

```
10 REM 四則演算（倍精度）
100 DEFDBL A-C
110 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
120 TM=FNT
130 FOR I=1 TO 10000
140   A=A+1
150   B=B-1
160   C=C*3
170   C=C/3
180 NEXT I
190 PRINT FNT-TM
200 END
```

プログラム 1-3-8　四則演算(単精度) ⑧

```
10 REM 四則演算 (単精度)
100 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
110 TM=FNT
120 FOR I=1 TO 10000
130   A=A+1
140   B=B-1
150   C=C*3
160   C=C/3
170 NEXT I
180 PRINT FNT-TM
190 END
```

プログラム 1-3-9　PEEK／POKE 文によるブロック転送 ⑨

```
10 REM ＢＬＯＣＫ転送(ＰＥＥＫ＆ＰＯＫＥ)
100 SCREEN 3:CLS 3
110 DEF SEG=&HA800
120 DIM A%(770)
130 FOR I=0 TO 100
140   LINE(I*3.19,0)-STEP(0,399),3
150   LINE(0,I*3.99)-STEP(319,0),3
160 NEXT I
170 DEF FNT=VAL(MID$(TIME$,4,2))*60+VAL(RIGHT$(TIME$,2))
180 TM=FNT
190 FOR I=0 TO 15999
200   POKE (I¥40)*80+I MOD 40 +40,PEEK((I¥40)*80+I MOD 40)
210 NEXT I
220 PRINT FNT-TM
230 END
```

	DISK-BASIC		MS-DOS BASIC		98VM	実行速度比(%)			
	8086 〔1〕	V30 〔2〕	8086 〔3〕	V30 〔5〕	V30 〔5〕	〔2〕/〔1〕	〔4〕/〔3〕	〔5〕/〔4〕	〔4〕/〔2〕
①	31.8	28.6	28.6	25.8	20.4	89.9	90.2	79.1	90.2
②	43.2	38.8	44.0	39.6	32.6	89.8	90.0	82.3	102.1
③	108.2	99.2	18.8	13.6	11.2	91.7	72.3	82.4	13.7
④	56.0	53.0	54.6	51.4	32.4	94.6	94.1	63.0	97.0
⑤	58.0	54.0	58.6	54.4	46.6	96.4	92.8	85.7	100.7
⑥	47.6	42.4	47.0	42.0	17.4	89.1	89.4	41.4	99.1
⑦	57.0	51.6	50.6	45.8	37.2	90.5	90.5	81.2	88.8
⑧	50.2	45.2	49.0	44.4	36.2	90.0	90.6	81.5	98.7
⑨	85.0	75.2	79.8	70.4	58.6	88.5	88.2	83.2	93.6
計	537.0	488.0	431.0	387.4	292.6	90.9	89.9	75.5	79.4

表 1-3-1　ベンチマークテスト結果

BASIC ROMの解析／新コマンド追加法

第2章

PC-9801シリーズは(XAを除く)，ROM内にN 88-BASIC(86)インタープリタを内蔵しており，それらのサブルーチンを利用してプログラムを開発することができます．この章では，それらの具体的な利用法を示します．

2-1　システムサブルーチンの解説

2-1-1　システムサブルーチン

割り込みベクタNo.0（エラーメッセージの表示①）

機　能

ALに設定されたエラーコードに対応するエラーメッセージを表示します．ON ERROR GOTOが実行されていなければエラーメッセージを表示後コマンド入力モード(ダイレクトモード)に戻ります．ON ERROR GOTOが実行されているとエラーメッセージを表示せずに，指定された行から実行を開始します．

入　力

内部割り込みコード：0C4H

DI ：0

AL：エラーコード

SS ：60 H

出　力

エラーメッセージが表示され表示入力モードに戻ります．表示されるエラーメッセージと指定されたエラーコードとの対応はPC-9801シリーズ各々の「BASICリファレンスマニュアル」に示されています．

割り込みベクタ No.1，2，3（エラーメッセージの表示②）

機　能

「エラーメッセージの表示①」のサブルーチンに当たります．

入　力

内部割り込みコード：0C4H

DI：01(DI：02，DI：03)

出　力

DI：01(エラーコード2「Syntax error」に対応)

DI：02(エラーコード5「Illegal function call」に対応)

DI：03(エラーコード13「Type mismatch」に対応)

割り込みベクタ No.4（エラーメッセージの表示③）

機　能

演算実行時のエラー(オーバーフロー，ゼロデバイド)は，エラーメッセージ表示後，処理を続行します．このような場合をリターンモードと呼びます．

入　力

内部割り込みコード：0C4H

DI ：4

AL：エラーコード(6または11)

SS ：60H

出　力

リターンモードでのエラーメッセージの表示は次のように略称メッセージの形式になります．

エラーコード6(オーバーフロー)：〝OV〞

エラーコード11(ゼロデバイド)：〝／0〞

割り込みベクタ No.5，6，7，48（倍精度実数型演算）

機　能

システム共通域のエリアを利用して，倍精度実数型演算を行います.

(BCDFC)←(BCADF)＋－＊／(BCDFC)

1416 H	1420 H	1416 H
8 バイト	8 バイト	8 バイト

注　意

データ型は演算項，被演算項，演算結果ともに倍精度型を示します.

入　力

DS／SS：60 H

内部割り込みコード：0 C 4 H

DI：5(加算)，6(乗算)，7(除算)，48(減算)

BCADF (1420 H-8 バイト)：被演算項(倍精度型)

BCDFC (1416 H-8 バイト)：演算項(倍精度型)

BCVAL (1414 H-1 バイト)：08 H

出　力

BCDFC(1416 H-8 バイト)：演算結果

注　意

演算の結果，オーバーフローが生じたとき，及び除算で BCDFC(1416 H)がゼロのとき，絶対値は最大になります.

割り込みベクタ No.8（小数点以下の値の切捨て）

機　能

倍精度実数型データの小数点以下の値を切り捨てることによって，整数型データの変換をします.

入　力

内部割り込みコード：0 C 4 H

DI：8

BCDFC(1416 H)：変換する倍精度型実数値

BCVAL(1414 H)：08 H

出　力

BCDFC(1416 H)：倍精度型実数の整数部分の値

BCVAL：08 H

注　意

DI：12との違いに注意してください．このシステムサブルーチンでは，BCVALの値をチェックしていませんが，BCDFCに倍精度型実数値を入れないと正常な値が返されないことがあります．

割り込みベクタ No.9，42（型変換：数値データ→倍精度実数型）

機　能

数値データ(整数型，単精度実数型，倍精度実数型)を倍精度実数型に型変換します．被変換データ，変換された結果はシステム共通域に設定します．

(BCDFC)←(BCFCL)141 AH，4バイト　：整数型または単精度実数型
　1416 H　(BCDFC)1416 H，8バイト　：倍精度実数型
　8バイト

入　力

内部割り込みコード：0C4H

DI：9

DS／SS：60 H

BCFCL(141 AH-4バイト)：整数値または単精度実数値

BCDFC(1416 H-8バイト)：倍精度実数値

BCVAL(1414 H-1バイト)：数値データの型……02 H：整数
　04 H：単精度実数
　08 H：倍精度実数

出　力

BCDFC(1416 H-8バイト)：変換した結果(倍精度実数型)

BCVAL(1414 H-1バイト)：08 H

注　意

入力時のBCVALに02，04，08以外の値を指定するとエラーになります．DI：42の場合もDI：9と同様の処理を行います．

割り込みベクタNo.10,11（オーバーフロー／ゼロ除算エラー処理）

機　能

数値データ演算においてオーバーフロー，またはゼロ除算した場合の後処理を行います．オーバーフローの場合は絶対値を最大値に設定し，エラーメッセージを表示します．ゼロで除算したときの後処理も同様です．

入　力

内部割り込みコード：0C4H

DI　：10(オーバーフローの後処理)またはDI：11(ゼロ除算の後処理)

AL：complemaent sign

SS　：60H

出　力

・演算結果格納エリアに最大値または最小値が設定されます(絶対値では最大値)．

・エラーメッセージが表示されます．

割り込みベクタNo.12（型変換：数値データ→整数型）

機　能

数値データ(整数型，単精度実数型，倍精度実数型)を整数型数値に変換します．被変換データ，変換された結果は，システム共通域に設定します．設定エリア，データ形式は5を参照して下さい．

入　力

DS／SS：60H

内部割り込みコード：0C4H

DI：12

BDFCL(141AH-4バイト)：整数値または単精度実数値

BCDFC(1416H-8バイト)：倍精度実数値

BCVAL(1414H-1バイト)：数値データの型……02H：整数

04H：単精度実数

08H：倍精度実数

出　力

BCFCL (141 AH-4 バイト)：変換した結果(整数型)

BCVAL(1414 H-1 バイト)　：02 H

注　意

変換される実数型数値の指数部が 90 H(2^16) 以上の場合はエラーとなります．

割り込みベクタ No.13（トークン抽出）

機　能

テキスト中のトークンを抽出します．テキストポインタ BCNTXP(6 EAH)が指示する文字から始まるトークンの情報を目的のレジスタにセットします．

入　力

内部割り込みコード：0 C 4 H

DI：13(ROM サブルーチンベクタエントリー番号)

DS／SS：60 H(テキストセグメントベースと同一)

BCNTXP(6 EAH)：トークン情報を得ようとするトークン

出　力

BX，AL，SI，DX はそれぞれ表 2-1-1 のような内容をもちます．

BCCTXP(6 E 8 H)：読み込んだトークンの先頭アドレス

BCNTXP(6 EAH)：読み込んだトークンの次のトークンの先頭アドレス

注　意

＜区切り文字コード＞

(……(28H)　左カッコ
) ……(29H)　右カッコ
, ……(2CH)　コンマ
: ……(3AH)　コロン
; ……(BBH)　セミコロン
　……(00H)　行の終わり

＜型コードと DL＞

02 H：整数　(INT)
03 H：文字列　(STR)
04 H：単精度実数(SNG)
05 H：漢字文字列(KANJI)
08 H：倍精度実数(DBL)

※(BX)＝0→変数名
　(BX)＝1→配列名

※文字列定数の場合，FACC 1 にはストリングディスクリプタのアドレスを格納します．

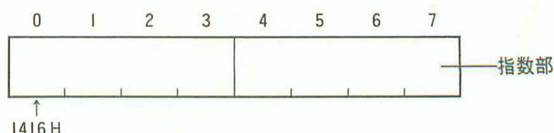


図 2-1-1　FACC 1

トークン＼レジスタ	BX	AL	SI	DX	その他
名前(左カッコなし)	0000H	名前の先頭文字	2番目の文字のアドレス	DL：型コード DH：名前の長さ－1	BCVAL (1414H) ←型コード
名前(左カッコあり)	0002H	同上	同上	同上	
関数名(FN<名前>)	0004H	同上	同上	同上	
数値定数	0006H	同上	不定	不定	値は FACC－1
文字定数	0008H	同上	不定	不定	値は FACC－1
USR関数(USRi)	000AH	不定	不定	関数番号	
組込み関数	000CH	不定	不定	組込み関数コード	
区切り	000EH	区切り文字コード	不定	不定	
・	0010H	不定	不定	行番号	
ラベル名	0012H	名前の先頭文字	2番目の文字のアドレス	DL：不定 DH：名前の長さ－1	
テキストアドレス	0014H	不定	不定	テキストアドレス	
行番号	0016H	不定	0EHのアドレス	行番号	
予約語	予約語コード (≧80)	不定	不定	不定	

表2-1-1 トークンタイプ

● FACCの内容

演算を行うときに使用する数値データの受渡しエリアをFACCと呼びます．演算項目(演算結果項目)，被演算項目用の2個をもち，それぞれFACC 1，FACC 2と呼びます．演算は次のように行います．

(演算項目)←(被演算項目)＋－＊／(演算項目)
FACC 1　　FACC 2　　　　　FACC 1

● FACCの形式と具体的なラベル(アドレス)

各データの型によって，使用する領域が異なり，それぞれラベル(アドレス)をもっています．エリアは8バイトからなり型によって使い分けます．

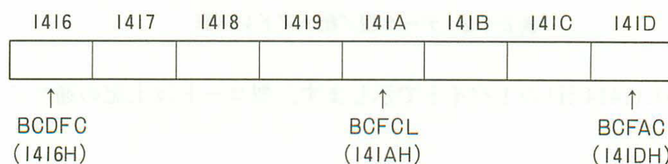


図2-1-2　FACC 1(演算項目，または演算結果項目)



図 2-1-3　FACC 2(被演算項目)

● データの型とデータ形式，使用する FACC

データの型 ＼ 桁・アドレス	BCDFC(1416H) BCADF(1420H) ↓				BCFCL(141AH) BCAFL(1423H) ↓		BCFAC(141DH) BCAFC(1427H) ↓	
・整数(INT) (02H)					数値			
・文字列(STR) (03H)					オフセットアドレス		セグメントベース	
・単精度実数(SNG) (04H)					仮数部			指数部
・漢字文字列(KNJ) (05H)					オフセットアドレス		セグメントベース	
・倍精度実数(DBL) (08H)	仮数部							指数部

表 2-1-2　データ型／桁・アドレス表

データの型はBCVAL(1414 H)の1バイトで示します．型コードは上記の通りです．

割り込みベクタ No.14（数式評価）

機　能

テキスト中の数式を評価します．テキストポインタ BCNTXP の指示する位置より，数式を評価し，その結果を所定の領域にセットします．

入　力

内部割り込みコード：0C4H

DI／SS：60 H

BCNTXP(オフセット 6 EAH，セグメントベース 60 H)：数式評価しようとするトークン

出　力

型の略称名	結果の型 (1414H) ECVAL	結果の値 BCDFC	結果の値 (1416H)	結果の値 (141AH) BCFCL	結果の値 (141DH) BCFAC
INT	02			整数	
STR	03			ストリングディスクリプタアドレス	セグメントベース
SNG	04			仮数部	指数部
KNJ	05			ストリングディスクリプタアドレス	セグメントベース
DBL	08	仮数部	仮数部	仮数部	指数部

表 2-1-3　出力条件

BCNTXP(6 EAH)：評価した数式の次のトークンのアドレス

割り込みベクタ No.15（ストリングデータエリアの通知）

機　能

文字列演算に使用された文字列を参照するために，その文字列の格納アドレスを獲得します．文字列演算が行われる場合には，使用された文字列定数・文字列式の演算結果のストリングディスクリプタがインタプリタ共通エリア(BCTSDA・142 AHから4バイトエントリ32個)に格納されています．このBCTSDAの32個のどのアドレスが必要なのかをBCTSDP(14 AAH)で指定します．



図2-1-4　アドレスの指定

入　力

内部割り込みコード：0C4H

DI：15

DS／SS：60H

BCTSDP(14 AAH)：求めるBCTSDAテーブルのエントリーアドレス

出　力

CX：文字列の長さ(CH＝00となる，入力条件と同じ)

DI ：文字列格納エリアの先頭オフセットアドレス

ES ：文字列格納エリアのセグメントベース

BX：BCTSDAテーブルのエントリーアドレス

割り込みベクタNo.16(テキストの内部形式情報を外部表現に変換)

機　能

内部表現をした1行目のテキストを外部表現に変換します．テキストをソース形式で表示したり，アスキーセーブをするときに用います．変換したデータはテキストセグメント内の指定したエリアに格納します．変換中に変換したデータが255バイトを超えると，超えた部分は捨てられます．

入　力

内部割り込みコード：0C4H

DI ：16

SI ：変換するテキストの先頭アドレス

BP ：アイテムアドレス(変換した文字のアドレスを知る必要があるアイテムを示します)

BX：変換した結果を格納する出力バッファの先頭アドレス

出　力

保証されるレジスタ：BX，CX，AX，DX以外のレジスタ

BP ：入力時のBPが指す変換前のテキストアイテムに対応した変換後の変換文字のアドレス

BX：変換したソースイメージの最後の変換アドレス+1

CX：変換した文字列長

注　意

BPが指すアイテムは，このアイテムに対応した変換文字のアドレスを必要とする場合に使用します．例えば，エラーがあるアイテムはソースイメージに変換した後で，そのソースイメージ上の位置を教える必要があります．このような場合にBPを使用します(アイテムとは，トークンにブランク項目を付加したものです)．

(アイテム項目)=(トークン項目)+(ブランク項目)

割り込みベクタ No.17 (行番号をバイナリ値に変換)

機　能

テキスト生成時において，行番号数字列をバイナリ値に変換します．次の条件に従って変換します．

・数字，スペース以外の文字が現れると変換は終ります．

・行番号の最大値(65529)を超えると，超えた直前の数字までの変換結果を返します．

・行番号が0(ゼロ)でもエラーにはなりません．

入　力

内部割り込みコード：0C4H

DI：17

SI：文字列の先頭アドレス(DS：セグメントベース)

出　力

保証されるレジスタ：AX，SI以外のレジスタ

AX：変換した結果(バイナリ値)

SI　：最後の数字のアドレス+1

割り込みベクタ No.18（ファイル番号，FCBアドレスのチェック）

機　能

BASICの「OPEN～#n」「INPUT #n～」文等のファイル番号を指定する文の数式を評価し，対応するFCBのアドレスを求めます．数式評価の結果，妥当なファイル番号でないと「Bad File Data」(エラーコード25)のエラーになります．

入　力

内部割り込みコード：0C4H

DI　：18

AL：処理条件

0	1	2	3	4	5	6	7
×	×	×	×	×	×	②	①

図2-1-5　処理条件

X……未使用

①……ファイル番号0の処理

0：エラーとします

1：エラーとしません

②……#の処理

0：読みとばします

1：エラーとします

BCNTXP(6EAH)：数式の先頭または#トークンの先頭アドレス

出　力

保証されるレジスタ：DS，SS

BCFCBAD(1538H)：FCBのアドレス

BX：FCBのアドレス(BCFCBADと同じ)

BCFNUMB(1536H)：ファイル番号

BCNTXP(6EAH)：数式の区切りトークンのアドレス

割り込みベクタ No.19（ファイルディスクリプタの解析）

機　能

ファイルディスクリプタの内容を解析し，それぞれの項目を所定のシステム共通域のエリアに格納します．

デバイス名　　　　　　　→ BCDVTP　(152 AH)
デバイスコード(装置番号)→ BCUNTNO (152 BH)
ファイル名　　　　　　　→ BCFNAME (152 CH)

デバイス名称	デバイス名(コード化)	装置番号
DISK	00	01～0AH
CMT	01	01, 02
COM	02	01
LPT	03	01
SCREEN	04	01
KEYBOARD	05	01

表 2-1-4　デバイス名／装置番号

ファイル名は6文字のファイル名と3文字の拡張子で表記します．

入　力

内部割り込みコード：0C4H

DI：19

BCNTXP(6EAH)：テキストポインタでディスクリプタを指す

出　力

保証されるレジスタ：BP

BCDVTP(152 AH)

BCUNTNO(152 BH)

BCFNAME(152 CH)

注　意

「Bad Drive number」(エラーコード70)のチェックを行います．

割り込みベクタ No.20（ファイルのデバイス名，装置番号の取得）

機　能

指定したFCBに対するデバイス名，装置番号を得ます.

入　力

内部割り込みコード：0C4H

DI：20

BCFCBAD(1538H)：求めるファイルのFCBアドレス

出　力

BCDVTP(152AH)：デバイス名(記号化した)……前項参照

BCUNTN0(152BH)：装置番号……前項参照

割り込みベクタ No.21（サブルーチンの呼び出し）

機　能

サブルーチン形式をしたN88-BASIC(86)インタープリタ上のプログラムから制御を移し，処理後，呼び出し元へ戻ります. ROM上のエントリポイントが明らかなサブルーチン形式のプログラムが使用できます.

入　力

内部割り込みコード：0C4H

DI：21

(0A00H～0A01H)：サブルーチンエントリポイント

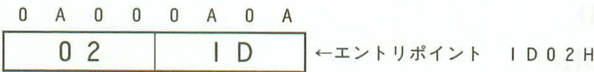


図2-1-6　入力条件(例)

DS：060H

呼び出すサブルーチンに必要なデータをセット

出　力

呼び出したサブルーチンの処理結果に従います.

割り込みベクタ No.22（データアドレスの取得……GETPTR）

機　能

変数名，または配列名のデータアドレス(オフセット及びセグメントベース)を得ます．指定された変数名，または配列名(GET TOKEN で得たトークン)に対して，

●変数名の場合

変数名が定義済みであれば，データ部のアドレスをBDPTR(154 EH)に格納します．未定義であれば変数名テーブルに名前を登録し，上記と同じ処理を行います．

●配列名の場合

添字をテキストポインタBCNTXP(6 EAH)の指示する位置より取り出し，データスタックにセットします(BPが格納エリアをポイントしています)．このセットしたアドレスをBCPTR(154 E)に格納します．

入　力

内部割り込みコード：0 C 4 H

DI ：22

AL：先頭の文字

DL：型

DH：(変数名の長さ)-1

SI ：2 番目の文字のアドレス(オフセット)

DS ：2 番目の文字，または添字テーブルアドレスのセグメントベース

BP ：添字のテーブルアドレス(オフセット)

SS ：添字のテーブルアドレスのセグメントベース

BCNTXP：配列名の場合は左カッコの次を指す

BCCDSP ：配列名の場合はデータスタックを指す

出　力

保証されるレジスタ：DX，SP，BP，CS，DS，SS

BCPTR：格納されている変数名または配列名のデータアドレス(オフセットとセグメントベース)

割り込みベクタ No.23（指定したデータを FACC に変換）

機　能

指定したデータを指定したデータ型に変換し，FACC に格納します．

入　力

内部割り込みコード：0C4H

DI ：23

SI ：変換するデータのオフセットアドレス

CX：変換するデータの長さ(バイト)

DL：変換するデータ型

BCTSDP／DS：文字列データの場合，現在の TSD へのポインタ

CL ：文字列データの場合，文字列の長さ(バイト)

DH：文字列データの場合，文字列のセグメントコード

出　力

FACC1：指定したデータ型に変換した結果が格納されているエリア(型ごとのデータエリアを参照)

BCVAL：変換したデータ型

BCTSDP／DS：文字列データのディスクリプタを格納した TSD へのポインタ

割り込みベクタ No.24（FACC の内容ストア）

機　能

BCPTR(154 EH)で指すエリアに FACC の内容をストアします．FACC の内容が文字列の場合はストリングエリアに文字列情報を移動し，データディスクリプタは BCPTR(154 EH)を指すエリアに生成されます．

入　力

内部割り込みコード：0C4H

DI：24

BCPTR(154 EH)：FACC を格納するアドレス(オフセット，セグメントベース)

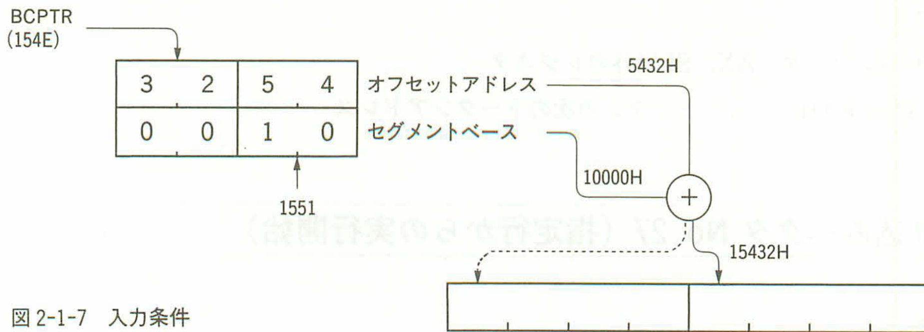


図 2-1-7　入力条件

BCVAL(1414 H)：FACC のデータ型

FACC を格納するエリアを定義(BCPTR(154 EH)で指す)

出　力

BCPTR(154 EH)で示したエリア：FACC の内容

割り込みベクタ No.25（データアドレスの取得……文字列）

機　能

文字列データのデータアドレスを取得します(No.17 を参照してください).

入　力

DI：25

他の条件は No.17 と同様です.

出　力

No.17 と同様です.

割り込みベクタ No.26（トークンのスキップ）

機　能

次のトークンが〝,〟ならば，それをスキップします．〝,〟ではない場合，〝Syntax error〟を表示します.

入　力

内部割り込みコード：0 C 4 H

DI：26

BCNTXP(6 EAH)：次のトークンアドレス

出　力

保証されるレジスタ：AX，SI以外のレジスタ

BCNTXP(6EAH)：“,”トークンの次のトークンアドレス

割り込みベクタ No.27（指定行からの実行開始）

機　能

テキストセグメント内にある指定した行から実行を開始します．

入　力

内部割り込みコード：0C4H

DI ：27

BX：実行する行の先頭アドレス

出　力

BCCLNH(6ECH)：実行中の先の先頭アドレス(を格納しています)

BCCLNO(6E4H)：実行中の行番号

BCCSTA(6EEH)：実行中の文の先頭アドレス

処　理

テキストエリア内のテキストの形式は次のとおりです．

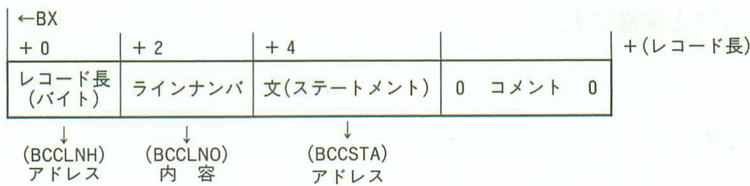


図2-1-8　テキストの形式

割り込みベクタ No.28（指定行の次の行から実行開始）

機　能

テキストセグメント内にある指定した行の次の行から実行を開始します．

入　力

内部割り込みコード：0C4H

DI：28

(BCCLNH(6ECH)：実行する行の前の行のアドレス)

出　力

No.29 と同様です.

処　理

No.29 と処理が異なる部分は次の通りです.

```
ENT 1:      EQU*
            MOV    BX,BCCLNH
            ADD    BX,[BX]
ENT 2:      CMP    [BX],OH
            JNE
```

No.29 は ENT 2 から制御を開始、No.30 は ENT 1 から制御を開始します.

割り込みベクタ No.29（数値定数を外部表現から内部表現に変換）

機　能

数値定数(整定数, 実定数, 倍精度定数, 16 進定数, および 8 進定数)を外部表現から内部表現に変換します.

入　力

内部割り込みコード：0C4H

DI：29

SI：文字列の先頭アドレス

出　力

BCFCL (1416 H)：定数の値

BCVAL(1414 H)：定数の型

SI ：処理した文字列の次のアドレス

DL：定数の型(BCVAL(1414 H)と同じ)

AX：定数値(整数の場合のみセットします)

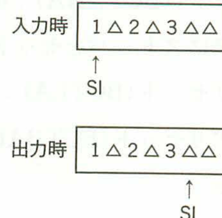


図 2-1-9　入力条件

※途中のブランクは読みとばしますが後の空白は読みとばしません(値は 123：BCFL).

割り込みベクタ No.30（行入力モードに移行）

機　能

今までの入力状態をリセットし，"OK"のメッセージを表示します．次にAUTOモードならば次の行番号を表示し，入力を待ちます．入力された行はテキストエリアに格納されます．AUTOモードでない場合は，入力された行をインタプリトするためにインタープリタに制御を渡します．このような行入力状態が続きます．

入　力

内部割り込みコード：0C4H

DI：30

出　力

特にありません．

割り込みベクタ No.31（テキスト編集のステータスリセット）

機　能

テキストの編集を行うために実行環境のリセットを行います．

入　力

内部割り込みコード：0C4H

DI：31

出　力

次の通りにリセットを行います．

- BCOPFG
- BCOPBS
- テキストストッパのセット((BCTXNA)：0)
- データスタックポインタにストッパをセット((BCDSAD-2)：0)
- ストリング格納エリアリセット(BCTLA)：(BCSTAD)
- シンボルテーブルクリアリセット(BCTBAD)：0100H
- ランダムシードリセット
- 型定義テーブルリセット
- DATA文使用エリアリセット　BCDATAP，(BCDATAL)：(BCTXAD)

・エラー情報エリアリセット(BCERRF)＝(BCLBTF)＝(BCERRH)＝(BCCONTH)＝(BCONERR)：0
・インタラプト制御データリセット(HELP OFF, STOP ON, GOSUB エントリーリセット)
・キーテーブルリセット

割り込みベクタNo.32(与えられたデータをUSING文字列で編集)

機 能

与えられたデータをUSING文字列で編集してバッファにセットします．

入 力

内部割り込みコード：0C4H
DI ：32
AL：1データごとの処理モード
　0：pre-process(search just-before of from)
　1：change data with using(on from)
　2：before process(search just-before of from, or terminal of using string)
BCVAL(1414H)：データ型
FACC1(141AH)：データ値
ES：出力バッファアドレス(セグメントベース)
DI：出力バッファアドレス(オフセットアドレス)
PRNUCT(184AH)：USING STRING カレントカラムカウンタ
PRNUAD(184CH)：USING STRING オフセットアドレス
PRNUSZ(1850H)：USING STRING サイズ
PRNUBS(184EH)：USING STRING セグメントベース

出 力

CX：出力ストリングの長さ

割り込みベクタNo.33(データの内部表現を外部表現に変換)

機 能

データの内部表現を外部表現である文字列に変換し，指定の出力バッファに出力します．データが文字列の場合は，その文字列が出力されます．データが数値の場合は，先頭文字に符号文字列(マイナス，またはスペース)をおき，次の文字列から絶対値の外部表現がおかれる形で出力されます．最終文字にはスペース及びタイプ文字(！，#)は付加しないものとします．

入 力

内部割り込みコード：0C4H

BCVAL(1414H)：データ型

FACC1(141AH)：データ値

DI：出力バッファのオフセットアドレス

ES：出力バッファのセグメントベース

出 力

AX：図2-1-10を参照

CX：出力文字列の長さ

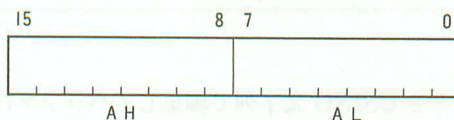


図2-1-10 出力条件

b15＝1の場合(LISTイメージでタイプ文字列が必要な場合)，AL：タイプ文字列

注 意

本コマンドは，DIを使用するためソフトウェア割り込みを使用できません．従って，FARCALLによって呼び出します．

割り込みベクタNo.34（スペースアイテムのスキップ）

機 能

スペースアイテムについて，テキストポインタをスキップさせます．現在のテキストポインタの値をスペースではないアイテムまでスキップさせます．

入 力

内部割り込みコード：0C4H

DI：34

BCNTXP(6EAH)：次のアイテム(項目)のアドレス

出 力

SI：スペースではないアイテムの次のアイテムアドレス(与えられたBCNTXPが指すテキストポインタから，最初に見つかったスペースでないアイテムのアドレス)

BCNTXP(6EAH)：スペースでないアイテムの次のアイテムアドレス

AX：スペースでないアイテムのアイテムコードナンバー

処 理

①1アイテムトークンを取得(GETITM)します．

②スペースかどうかの判定を行い，スペースならば①へ，スペースでなければリターンします．

割り込みベクタNo.35(文終端—EOS＝end of statements—の評価)

機　能

トークンが文の終端であるかどうかをチェックし，チェックした結果を返します．

入　力

内部割り込みコード：0C4H

DI：35

SI：テキストアイテムポインタ

出　力

C-FLAG……0：EOS(end of statements)

　　　　　1：NOT-EOS

処　理

```
cmpib  #si,00h……ラインの終わり(EOL)かどうかを判別
je     ret1
cmpib  #si,3ah……「：」マルチステートメントの区切り記号かどうかを判別
je     ret1
stc            ……EOSではないので，C-FLAGをONにする
ret1:  ret
```

割り込みベクタ No.36, 50, 51(バイナリー数値を文字列数値に変換)

機　能

バイナリーで表現されている数値を文字列の数値に変換します．変換の方法に関しては次の通りです．

①2バイトからなるバイナリーで表現された数値を8進数文字列に変換します．

②2バイトからなるバイナリーで表現された数値を16進数文字列に変換します．

③FACC上の数値データを10進数文字列に変換します．

ただし，変換後の文字列は，

①先頭に0が付きません．ただし，バイナリーが0ならば0と変換します．

②&,&H,&Oは出力しません．

入　力

内部割り込みコード：0C4H

DI……36(10進数文字列変換)……①

50(8進数文字列変換)…… ②

51(16進数文字列変換)……③

AX：数値データ(2バイト・バイナリー)……②③

BX：出力バッファのアドレス(オフセット)

DS：出力バッファのアドレス(セグメントベース　ただし，①の場合はEVEN BOURDORY)

BCVAL(1414H)：数値データ型　……①

FACC(1416H)：数値データ　……①

出力データバッファ(DS／BSでポイント)の定義

出　力

CX：文字列の長さ

出力データバッファ(DS／BSでポイントされています)：変換結果

注　意

10進数文字列変換の結果は次のような形をしています．

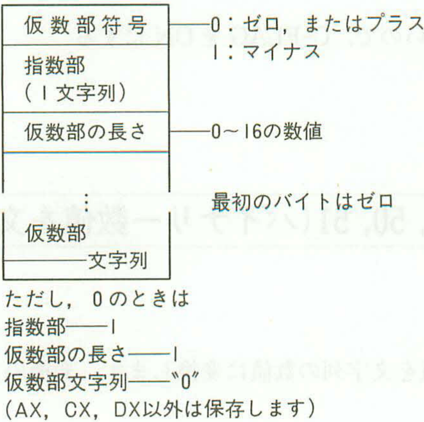


図2-1-11　変換結果

また，DI：33(データを10進文字列に変換する)は入力パラメータをDIにセットせねばならないため，機能的にはDI：36とほぼ共通にも関わらず，ソフトウェア割り込みを使用できません．

割り込みベクタ No.37（データ出力）

機　能

指定した物理装置にデータを出力します(PRINT OUTに対応しています)．

入 力

内部割り込みコード：0C4H

DI：37

PRNDV(1840 H)：出力装置タイプ……04 H：DISP
03 H：LP
その他：ファイル

PRNBAD(1842 H)：出力バッファのオフセットアドレス

PRNBSZ (1844 H)：出力バッファの長さ(バイト)……最大255(FFH)

DS：出力バッファのセグメントベース

出力バッファの定義

CX：出力文字列の長さ

出 力

保証されるレジスタ：AX，BX以外のレジスタ

処 理

・CX=0ならリターン.
・PRNDV=04 Hならば，(CX)だけディスプレイに出力します(¥IDSPLY).
・PRNDV=03 Hならば，出力バッファ分をプリンタへ出力します(¥ILPUTL).
・PRNDV=その他ならば，ファイルとみなして出力バッファ全体を出力します(¥IPUTL).

割り込みベクタ No.38，39，40（エラーメッセージの表示）

機 能

エラーメッセージの表示を行います.

入 力

内部割り込みコード：0C4H

DI……38(DI：1と同じ)
39(DI：3と同じ)
40(DI：2と同じ)

出 力

DI：38(エラーコード2「Syntax error」に対応)

DI：40(エラーコード5「Illegal function call」に対応)

DI：39(エラーコード13「Type mismatch」に対応)

割り込みベクタ No.41（数値データを単精度実数型データに変換）

機　能

数値データを単精度実数型データに変換します．

入　力

内部割り込みコード：0C4H

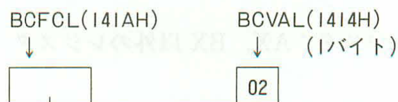
DI：41

BCVAL(1414 H)：数値型

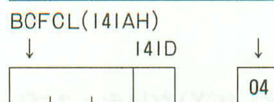
BCFCL (141 AH)：整数または単精度実数型データ(被変換データ)

BCDFC (1416 H)：倍精度実数型データ(被変換データ)

① 整数型データの場合



② 単精度実数型データの場合



③ 倍精度実数型データの場合



図2-1-12　入力条件

出　力

BCVAL(1414 H)：04 H

BCFCL (141 AH)：変換した結果(単精度実数型データ)

割り込みベクタ No.43, 44, 45, 46（単精度実数型データの演算）

機　能

単精度実数型データ間の演算を行います．

(結果)←(被演算項)＋(演算項)

(BCFCL)(BX)　(BCFCL)

入　力

内部割り込みコード：0C4H

DI……43(加算)

44(減算)

45(乗算)

46(除算)

BX：被演算項のオフセットアドレス

DS：被演算項のセグメントベース

BCFCL (141AH)：演算項

BCVAL(1414H)：データ型(04H)

出　力

BCFCL (141AH)：演算結果

BCVAL(1414H)：データ型(04H)

注　意

オーバーフローが生じたとき，及び除算でBCFCLがゼロの場合は絶対値を最大とします．

割り込みベクタ No.47，49（実数値の比較）

機　能

2つの単精度実数データ間，または2つの倍精度実数データ間の比較を行い，比較の結果を得ます．

入　力

内部割り込みコード：0C4H

DI……47(単精度実数型データ間の比較)

49(倍精度実数型データ間の比較)

BCVAL(1414H)：データ型(04Hまたは08H)

BCFCL(1416H)：右辺単精度実数型データ(DI：47の場合)

BCAFL(1420H)：左辺単精度実数型データ(DI：47の場合)

BCDFC(1416H)：右辺倍精度実数型データ(DI：49の場合)

BCADF(1420H)：左辺倍精度実数型データ(DI：49の場合)

出　力

保証されるレジスタ：AX，BX，CX，ES 以外のレジスタ

BCVAL(1414 H)：整数データ型(02 H)

BCFCL(1416 H)……0(左辺＝右辺)

1(左辺＜右辺)

-1(左辺＞右辺)

割り込みベクタ No.52（バイナリー数値を行番号文字列に変換）

機　能

バイナリー数値で与えられた数値を行番号文字列に変換し，指示されたバッファにセットします.

入　力

内部割り込みコード：0C4H

DI ：52

AX：バイナリー形式の行番号

BX：出力バッファアドレス

変換データ格納エリア(BX でポイントされます)

出　力

CX：文字列の長さ

変換データ格納エリア：変換データ

注　意

AX，CX，DX レジスタは破壊されます.

用　途

行番号を表示するトレース，AUTO 実行，LIST コマンドの処理に使用します.

割り込みベクタ No.53（テキストアドレスの行番号を書き換え）

機　能

テキスト中の飛び先などの実アドレスをすべて行番号に書き換えます.

入　力

内部割り込みコード：0C4H

DI：53

6D6H：0

出　力

6D6H：変換された結果

注　意

飛び先などの実アドレスとは，内部表現の0DH 2バイト数字のことです．

割り込みベクタ No.54（テキストから1項目抽出）

機　能

指定されたテキストアドレスの項目を記号化して抽出します．

入　力

内部割り込みコード：0C4H

DI：55

6EAH／6EBH：テキストアドレス

出　力

AL：抽出された項目の種別

AL	意　　味
0	00H(Null)，REM文，文のおわり
1	0AH(LF)，"(ダブルクォート)内をスキップします
2	0BH×××× 8進数
3	01H～09Hで表された空白
4	0CH×××× 16進数
5	0DH×××× アドレス
6	0EH×××× 行番号
7	0FH×× 10～255の整数
8	漢字シフト
9	2バイト整数
A	4バイト整数
B	8バイト整数
C	変数名
D	ステートメント
E	関数

表2-1-5　ALへのリターン値

6EAH／6EBH：次の項目のテキストアドレス

割り込みベクタ No.55，56（指定行番号を持つテキストアドレスの取得）

機　能

指定された行番号を持つ文のテキストアドレスを返します．

入　力

内部割り込みコード：0C4H

DI：55

6EAH／6EBH：行番号の先頭

出　力

BX：テキストの先頭アドレス

割り込みベクタ No.57（テキストのサーチ）

機　能

AXにより指定された行番号の存在，その行番号以降のテキストの存在を確認します．

入　力

内部割り込みコード：0C4H

DI：57

AX：行番号

出　力

CF：指定された行番号のテキストの存在の有無

ZF：指定された行番号以降のテキストの存在の有無

BX：テキストを検出した行の先頭アドレス

注　意

AX，BX，CXレジスタは破壊されます．

割り込みベクタ No.58（数式の存在チェック）

機　能

指定したテキストポインタの位置に，数式が格納されているかどうかをチェックします．

入 力

内部割り込みコード：0 C 4 H

DI：58

BCNTXP(6 EAH)：テキストポインタ

出 力

CF：数式の存在の有無

FACC 1 (141 AH)：結果

BCVAL (1414 H) ：結果の型……02 H：整数
04 H：単精度型実数
06 H：倍精度型実数

BCNTXP(6 EAH)：数式の次のトークンの先頭アドレス

注 意

数式のチェック中にエラーが発生した場合，シンタックスエラーとなりインタープリタに制御を移します．

割り込みベクタ No.59（CRT へ表示）

機 能

BUFF 2 に格納された CX の値によって与えられた長さの文字列を CRT に出力します．文字列の中にコントロールコードが含まれていた場合，コントロールコードとしてカーソルの移動等を行います．

入 力

内部割り込みコード：0 C 4 H

DI ：59

CX：文字列の長さ

BUFF 2 (202 H)：文字列(256 バイトまで)

出 力

特にありません．

注 意

0060 H：154 CH の内容が 00 H でなければ，ターミナルモードだと判断し，RS-232 C への結果出力などのリモート BASIC プロトコルの処理を行います．

0060 H：154 BH の格納されている内容が 01 H の場合は，CRT への出力が禁止されたとみなして，出力を行わずにリターンします．

割り込みベクタ No.60（CRTへのライン出力……無条件）

機　能

DI：59の場合との違いは，「注意」に書かれた条件に関わらずCRTに出力する点で，他の機能については同様です．勿論，コントロールコードが含まれている場合もコントロールコードとみなしてカーソルの移動等を行います．

入　力

内部割り込みコード：0C4H

DI　：60

CX：文字列の長さ

BUFF 2(202 H)：文字列

出　力

特にありません．

割り込みベクタ No.61（CRTへ1文字表示）

機　能

ALレジスタに格納された1バイトのデータ(キャラクタコード)をCRTに出力します．1バイトのデータがコントロールコードであればコントロールコードとみなし，カーソルの移動などを行います．

入　力

内部割り込みコード：0C4H

DI　：61

AL：キャラクタコード

出　力

特にありません．

割り込みベクタ No.62(カーソルのリセット)

機　能

カーソル位置を最左桁に移動させます．処理以前にカーソル位置が最左桁である場合はなにも処理をしません．BASICモードであるかターミナルモードであるかを判断した上で，CR+LF(キャ

リッジリターン＋ラインフィード(0 DH＋0 AH))を出力します．もしくは，ターミナルモードの場合は，RS-232 C の制御を行います．

入　力

内部割り込みコード：0 C 4 H

DI：62

出　力

特にありません．

注　意

AX，CX，SI，DI レジスタが破壊されます．

割り込みベクタ No.63（改行）

機　能

基本的に DI：62 の場合と同様の処理を行います．ただし，現在のカーソル位置が最左桁であっても，改行を行います．

入　力

内部割り込みコード：0 C 4 H

DI：63

出　力

特にありません．

注　意

AX，CX，SI，DI レジスタが破壊されます．

割り込みベクタ No.64（CRT に対する改行）

機　能

DI：63 の場合とは異なり，CRT に対してのみ改行コードを出力します．カーソル位置が最下行の場合は，テキスト画面のスクロールアップも行います．

入　力

内部割り込みコード：0 C 4 H

DI：64

出　力

特にありません．

割り込みベクタ No.65（CRT に対するカーソルリセット）

機　能

DI：64 の場合と異なり，CRT に対してのみカーソルのリセットを行います．DI：64 との違いはカーソル位置が処理実行以前に既に最左桁の場合，何も処理を行わないことです．

入　力

内部割り込みコード：0 C 4 H

DI：65

出　力

特にありません．

割り込みベクタ No.66（符合なし整数化）

機　能

FACC の内容を整数に変換します．

入　力

内部割り込みコード：0 C 4 H

DI：66

出　力

FACC：整数化された FACC の内容

割り込みベクタ No.67（配列変数の添字を評価）

機　能

配列変数の添字が適当であるかどうかを調べます．不適当である場合は〝Subscript out of range〃エラーを出力します．

入　力

内部割り込みコード：0C4H

DI：67

BCNTXP(6EAH)：添字の先頭アドレス

出　力

BP：添字に対応する配列変数が格納されているオフセットアドレス

BCNTXP：評価した添字の次のテキストポインタ

割り込みベクタ No.68（ガベージコレクション）

機　能

ストリング格納エリアのガベージコレクションを行います．

入　力

内部割り込みコード：0C4H

DI：68

出　力

特にありません．

割り込みベクタ No.69（無符号整数を単精度実数型に変換）

機　能

FACC1(141AH)に格納された無符号整数を単精度実数に変換して戻します．BCVAL(1414H)に04Hを格納し，FACC1のデータが単精度実数であることを示します．

入　力

内部割り込みコード：0C4H

DI：69

FACC1(141AH)：無符号整数

出　力

FACC1(141AH)：単精度実数

BCVAL(1414H)：04H(FACC1に格納されたデータが単精度実数であることを示します)

注　意

AX，BX，CX，DX，BP，SI，DIレジスタは破壊されます．

割り込みベクタ No.70（〝in〟と行番号の出力）

機　能

CRT の現在のカーソル位置に，〝in〟と表示し AX に格納された行番号のデータを無符号の 10 進整数として表示した後，改行とビープを鳴らす処理を行います．エラー発生後の処理，またはストップキーによるプログラムの停止による処理を行います．

入　力

内部割り込みコード：0C4H

DI　：70

DS　：60 H

BX：BUFF 2 の先頭アドレス(202 H)

AX：行番号

出　力

特にありません．

注　意

AX レジスタの値を FFFFH にした場合，〝in〟と行番号の出力は行わずに改行コードの出力とビープを鳴らす処理を行うだけです．

割り込みベクタ No.71（文の読みとばし）

機　能

テキストポインタに示される文を読みとばし，テキストポインタを進めます．

入　力

内部割り込みコード：0C4H

DI：71

BCNTXP(06 EAH)：読みとばす文の先頭アドレス

出　力

BCNTXP(06 EAH)：次の文が格納されているアドレス

割り込みベクタ No.72（データの読みとばし）

機　能

テキストポインタに示されるデータを読みとばし，テキストポインタを進めます．

入　力

内部割り込みコード：0C4H

DI：72

BCNTXP(06EAH)：読みとばすデータの先頭アドレス

出　力

BCNTXP(06EAH)：次の文が格納されているアドレス

割り込みベクタ No.73（文字列定数の評価）

機　能

文字列定数の格納されている先頭アドレスと終了アドレス，文字列定数の長さを調べます．

入　力

内部割り込みコード：0C4H

DI：73

DF：0

SI：文字列定数が格納されている先頭アドレス

出　力

DI：文字列定数が格納されている先頭アドレス

SI：文字列定数が格納されている終了アドレス+1

CX：文字列定数の長さ

割り込みベクタ No.74（数式の評価および整数化）

機　能

テキストポインタが示すアドレスに格納されている数式を評価した後，演算結果の小数点以下を四捨五入しFACC1に格納します．評価(演算)中に文法的な誤りがあった場合，〝Syntax error〟を，数式の範囲が整数型で表現できる範囲を超えた場合〝Overflow〟を出力しインタープリタに制御を戻します．

入　力

内部割り込みコード：0C4H

DI：74

BCNTXP(06EAH)：数式が格納されているアドレス

出　力

FACC1(141AH)：整数化された数式

BCVAL(1414H)：02H

BCNTXP(06EAH)：評価した数式の次のトークンが格納されているアドレス

注　意

"ON ERROR GOTO"による，エラー発生後の処理が指定されている場合，メッセージの出力を行わず，実行を指定されている行番号へ処理が移ります．

割り込みベクタ No.75（キーボード・センス）

機　能

キーボードの押下状態をリアルタイムで調べます．

入　力

内部割り込みコード：0C4H

DI：75

出　力

(052AH)：グループ0の押下状態

(052BH)：グループ1の押下状態

(052CH)：グループ2の押下状態

(052DH)：グループ3の押下状態

(052EH)：グループ4の押下状態

(052FH)：グループ5の押下状態

(0530H)：グループ6の押下状態

(0531H)：グループ7の押下状態

(0532H)：グループ8の押下状態

(0533H)：グループ9の押下状態

(0534H)：グループ10の押下状態

(0535H)：グループ11の押下状態

(0536H)：グループ12の押下状態

(0537H)：グループ13の押下状態

(0538H)：グループ14の押下状態

(0539H)：グループ15の押下状態

注　意

AX，BX，CX，DX，BP，SI，DI，ESレジスタが破壊されます．

ビット キーコードグループ	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7
0	ESC	! 1 ヌ	" 2 フ	# ァ 3 ア	$ ゥ 4 ウ	% ェ 5 エ	& ォ 6 オ	' ャ 7 ヤ
1	(ュ 8 ユ	) ョ 9 ヨ	ヲ 0 ワ	＝ －ホ	∧へ	: ¥ー	BS	TAB
2	Q タ	W テ	ィ ヨ イ	R ス	T カ	Y ン	U ナ	エニ
3	O ラ	P セ	~ @ ゛	{ 「 [゜		A チ	S ト	D シ
4	F ハ	G キ	H ワ	J マ	K ノ	L リ	＋ ; レ	* : ケ
5	} 」 コ ム	ッ Z ツ	X サ	C ソ	V ヒ	B コ	N ミ	M モ
6	<, , ネ	>○ ● ル	? ・ ／メ	ー ロ	SPACE	XFER	ROLL UP	ROLL DOWN
7	INS	DEL	↑	←	→	↓	HOME CLR	HELP
8	－	／	7	8	9	*	4	5
9	6	＋	1	2	3	＝	0	,
A	・	NFER						
B								
C	STOP	COPY	f・1	f・2	f・3	f・4	f・5	f・6
D	f・7	f・8	f・9	f・10				
E	SHIFT	CAPS	カナ	GRAPH	CTRL			
F								

表 2-1-6　キーボード・グループ一覧表

割り込みベクタ No.76（COM，PEN 割り込みのセンス）

機　能

RS-232 C やライトペン割り込みをセンスし，フラグを立てます．

入　力

内部割り込みコード：0 C 4 H

DI：76

割り込みベクタ No.77（1行トランスレート）

機　能

アスキーコードで書かれた１行のテキストをバイナリ表現に変換します.

入　力

内部割り込みコード：０Ｃ４Ｈ

DI ：77

CX：テキストの文字数

1406 H～：アスキーコードによるテキスト

出　力

テキスト内に出力されます.

割り込みベクタ No.78（メモリ・スイッチの状態調査）

機　能

メモリ・スイッチの状態を調査します.

入　力

内部割り込みコード：０Ｃ４Ｈ

DI ：78

BX：メモリ・スイッチのオフセットアドレス

出　力

AL：メモリ・スイッチの状態

論理スイッチ名	物理アドレス	ベース：オフセット
メモリ・スイッチSW1	Ａ３ＦＥ２Ｈ番地	Ａ３Ｆ０Ｈ：００Ｅ２Ｈ番地
メモリ・スイッチSW2	Ａ３ＦＥ６Ｈ番地	Ａ３Ｆ０Ｈ：００Ｅ６Ｈ番地
メモリ・スイッチSW3	Ａ３ＦＥＡＨ番地	Ａ３Ｆ０Ｈ：００ＥＡＨ番地
メモリ・スイッチSW4	Ａ３ＦＥＥＨ番地	Ａ３Ｆ０Ｈ：００ＥＥＨ番地
メモリ・スイッチSW5	Ａ３ＦＦ２Ｈ番地	Ａ３Ｆ０Ｈ：００Ｆ２Ｈ番地
メモリ・スイッチSW6	Ａ３ＦＦ６Ｈ番地	Ａ３Ｆ０Ｈ：００Ｆ６Ｈ番地

表 2-1-7　メモリ・スイッチとオフセットアドレスの相対表

割り込みベクタ No.79（RAMの実装状態調査）

機　能

その時点での"DEF SEG"文により指定されたアドレス(0060：0750，0751に格納されている)をセグメントベースとして，BXレジスタの内容をオフセットとする物理アドレス上にRAMが実際に実装されているかどうかを調べます．実装されている場合は，何も行わずに戻りますが，実装されていない場合"Illegal function call"を出力してインタープリタに制御を移します．

入　力

内部割り込みコード：0C4H

DI ：79

AX：オフセットアドレス

出　力

特にありません．

割り込みベクタ No.80（ストリング・エリアの確保）

機　能

指定された長さの文字列領域をストリング・エリアに確保します．空きエリアが確保できない場合は"Out of string space"を出力してインタープリタに制御を移します．

入　力

内部割り込みコード：0C4H

DI ：80

CX：文字列の長さ

出　力

6B4H/6B5H：ストリング・エリアのポインタ

割り込みベクタ No.81（キーワードのサーチ）

機　能

指定されたテキストアドレスから，キーワードをサーチします．

入　力

内部割り込みコード：0C4H

DI　：81

AL：キーワード

6B4H／6B5H：テキストアドレス

出　力

AL：0(見つからなかった場合)

6B4H／6B5H：キーワードのテキストアドレス

割り込みベクタ No.82（キーボードより1行入力）

機　能

キーボードからの入力をバッファに格納します．入力された文字列の終りにはエンドマークとして0DH，0AHが付け加えられます．

入　力

内部割り込みコード：0C4H

DI：82

出　力

202H～302H：入力バッファ

2-1-2　システムサブルーチンの使用法

(1)割り込みベクタによる利用法

N88-BASIC(86)のROMに内蔵されている87種類のサブルーチンを，割り込みによって利用する方法を示します．

(2)INT　C4(ソフトウェア割り込み)による利用

割り込みベクタを指定するやり方の他に，このようなシステムサブルーチンを利用するための特別な処理ルーチンが用意されています．

※参考プログラム(文字列をテキスト画面に出力する)

● N 88-DISKBASIC インタプリタ上の BASIC プログラム(2-1-2 A)と同じ動作をするシステムコールを使用した機械語ルーチンを(2-1-2 B)に示します．なお，プログラムは説明の都合上，BASIC の DATA 文をメモリに書き込むようにしました．注釈文に書かれている通り，機械語モニタのアセンブラでこのルーチンを記述することもできます．

プログラム 2-1-2 A　BASIC インタプリタを利用した例

```
100 '*************************************************************************
110 '*  2－1－2  サンプルプログラム（1）  1986.5.8 (C)Copyright DMSC
120 '*************************************************************************
130 '
140  RESTORE 210
150  READ  A$
160  FOR I=1 TO 10
170     PRINT CHR$(VAL("&H"+A$));
180  NEXT I
190  END
200 '
210 DATA 54                              :' T
```

プログラム 2-1-2 B　システムコールを利用した例

```
100 '******************************************************************
110 '           2－1－2B  サンプルプログラム（1）
120 '  update 1986.05.08 copyright (C) DMSC ( Use N88-DISKBASIC )
130 '******************************************************************
140 '
150 CLEAR ,&H1F00 : DEF SEG=&H1F00
160 '
170 ' CODE
180 FOR I=0 TO &H1F
190   READ A$ : POKE I,VAL("&H"+A$)
200 NEXT I
210 '
220 ' DATA
230 I=&H30
240   READ A$ : POKE I,VAL("&H"+A$)
250 '
260 '
270 A=0 : CALL A
280 '
290 END
300 ' CODE
310 DATA B8,60,00               ' 0000  MOV     AX,0060
320 DATA 8E,D8                  ' 0003  MOV     DS,AX   ;DS=60H
330 DATA 8E,D0                  ' 0005  MOV     SS,AX   ;SS=60H
340 DATA B9,0A,00               ' 0007  MOV     CX,000A
350 DATA BB,30,00               ' 000A  MOV     BX,0030 ;データは30Hから
360 DATA 31,C0                  ' 000D  XOR     AX,AX   ;AX=0
370 DATA BF,3D,00               ' 000F  MOV     DI,003D ;DI=3DH
380 DATA 31,E4                  ' 0012  XOR     SP,SP
390 DATA 2E                     ' 0014  CS:
400 DATA 8A,07                  ' 0015  MOV     AL,[BX]
```

```
410 DATA CD,C4                  ' 0017  INT     C4        ;テキストに表示
420 DATA E2,F2                  ' 0018  LOOP    000D
430 DATA BF,1E,00               ' 001A  MOV     DI,001E ;ＤＩ＝１Ｅ
440 DATA CD,C4                  ' 001C  INT     C4        ;インタプリタに戻る
450 '
460 ' DATA
470 DATA 54                     ' "T"
480 '
490 '             ******* programed by T.YASUI ********
```

2-1-3 ROM内ルーチンのCALLによる利用

2-1-2のように割り込みを使用すれば，比較的安全にこれらのROMルーチンを利用できますが，もう一つの方法として直接番地を参照する方法もあります．しかし，この方法がどうして危険なのかと申しますと，機種やROMのバージョンによって，其の絶対番地に差があるため，場合によっては，とんでもない番地を参照しプログラムが暴走する可能性が生じるということです．

そのような不都合が出ないように，個々の機種の絶対番地を出力するプログラムを紹介します．

プログラム 2-1-3　ベクタアドレス検索プログラム

```
1000 '* bector.bas ************************************************************
1010 '          ベクタアドレス検索プログラム for Disk-Basic 1986.6.17
1020 '*************************************************************************
1030 '
1040 DEF SEG = &H60
1050 OFFSET  = PEEK(&HA03)*256+PEEK(&HA02)
1060 DEF SEG = PEEK(&HA05)*256+PEEK(&HA04)
1070 '
1080 PRINT "出力装置は? (1)CRT: (2)PRN: ";
1090  A$=INPUT$(1)
1100  PRINT " "
1110  IF A$<>"2" THEN DEV$="SCRN:" ELSE DEV$="LPT:"
1120 '
1130  OPEN DEV$ FOR OUTPUT AS #1
1140 '
1150          PRINT #1,"No.","ｱﾄﾞﾚｽ","ｻﾌﾞﾙｰﾁﾝ ｱﾄﾞﾚｽ"
1160 '
1170          FOR I = 0 TO 86
1180                  ADDRESS = PEEK(OFFSET+I*2+1)*256+PEEK(OFFSET+I*2)
1190                  PRINT #1,I,HEX$(OFFSET+I*2),
1200                  PRINT #1,HEX$(ADDRESS)
1210          NEXT I
1220 '
1230  CLOSE #1
1240 END
1250 '        update 1986.06.05 copyright (C) DMSC
```

2-2 ディスクシステム

2-2-1 DISK BIOS 共通情報

DISK BIOS コマンドの一般形式は次のとおりです.

●内部割り込みコード

1 MB, 640 KB, 320 KB, HDD(PC-9801 の HDD を除く) : INT 1 BH

PC-9801 の HDD : INT 0B1H(E/F でも動作可能)

●レジスタへの入力

AH : BIOS コマンド識別コード(コマンド一覧表参照)

AL : デバイスタイプ識別コード(DA : Drive Address)およびユニット番号 (UA : Unit Addres)

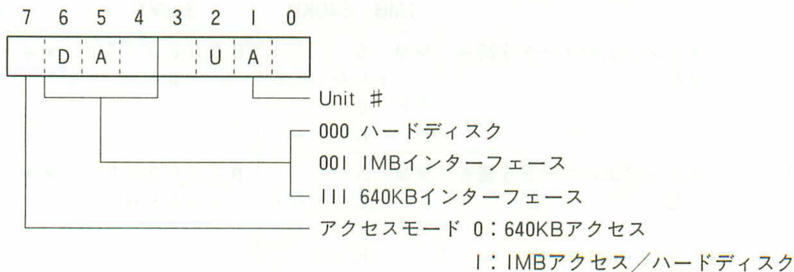


図 2-2-1 AL レジスタの構造

BX : データ長(DTL)

CX : セクタ長(N)

CL : シリンダ番号(C)

DH : ヘッド番号(H)

DL : セクタ番号(R)

ES : データバッファ領域先頭アドレス(セグメントアドレス)

BP : データバッファ領域先頭アドレス(オフセットアドレス)

●レジスタへの出力

CF : 終了条件……0 : 正常終了
1 : 異常終了

AH : ステータス情報(ステータス情報一覧表参照)

出力情報として使用されるレジスタ, フラグ以外は全て保証されます. システム共通域中の DISK_RESULT(564 H～583 H)の 8 バイトエントリ, または, F 2 DD_RESULT(5 D 0 H～5 DFH)の 16 バイトエントリに FDC からのリザルトステータスを格納します.

<BIOS コマンド使用時の諸注意>

●コマンドコードではないコードを指定した時

コマンドコードはAHレジスタに指定しますが，コマンドコードではないものをAHレジスタに指定したとき，レジスタCFには0が返され，正常終了とされます．

●データバッファ領域の定義について

DMAを使用する1MB，640KB，HDDの場合，データバッファはバンクにまたがって定義することはできません．また，データバッファの大きさは物理セクタの整数倍でなければなりません．両方が満たされていない場合，DB(DMA boundary)エラーとなります．

2-2-2 BIOSコマンド

コマンド	機　　能	コマンドコード(AHレジスタ)		
		1MB／640KB	320KB	ハードディスク
RFAD DATA	ディスク上のデータを読み込む	MM　S rE0110 TF　E K	00000110 (＝06H)	**r*0110
WRITE DATA	ディスク上にデータを書き込む	MM　S rE0101 TF　E K	00000101 (＝05H)	**r*0101
SEEK	指定された物理番号のシリンダヘアームを移動し，ヘッドを選択する	00r10000 (＝10H)		
RECALIBRATE	物理番号のシリンダ0ヘアームを移動する	00r00111 (＝07H)		**r*0111
FORMAT DRIVE	ディスクをフォーマットする		00001101 (＝0CH)	1*r*1101
INITIALIZE	コントローラの初期化を行う	00000011 (＝03H)	00000011 (＝03H)	00000011 (＝03H)
VERIFY	ディスク上のデータを読み取る（メモリには転送しない）	MM　S rE0001 TF　E K	00000001 (＝01H)	**r*0001
SENCE	デバイスの状態，属性を取得する	00000100 (＝04H)	00000100 (＝04H)	00000100 (＝04H)
READ ID	トラック上のエラーのないIDを読み取る	M　S 0　rE1010 F　E K		
WRITE DELETED DATA	DDAM(Deleted Data Address Mark)付きデータを書き込む	MM　S rE1001 TF　E K		

FORMAT TRACK	1トラック分のセクタフォーマットを行う	M S 0 r E 1 1 0 1 F E K		0 * r * 1 1 0 1
ASSIGN ALTER NATE TRACK	代替トラックを指定する			0 0 0 0 1 0 0 0 (＝0 8 H)
FORMAT BAD TRACK	不良トラックに代替トラックを割り当てる			0 0 0 0 1 0 1 1 (＝0 B H)
SET OPERATION MODE	片面／両面アクセスモードを指定する		0 0 0 0 1 1 1 0 (＝0 E H)	
READ DELETED DATA	DDAM付きデータを読み取る	M M S r E 1 1 0 0 T F E K		
READ DIAGNOSTIC	ID／DATA部のエラーが検出されても読み取りを続行する	M S 0 r E 0 0 1 0 F E K		
RETRACT	ヘッドを不使用シリンダへ移動する			* * r * 1 1 1 1

表 2-2-1 BIOS コマンド一覧

MT　1：マルチトラック指定
　　0：シングルトラック
MF　1：倍密度指定
　　0：単密度
SEEK　1：SEEK 動作を行う
　　0：SEEK 動作を行わない
r　1：リトライ指定
　　0：リトライなし
*　0，1のどちらでもよい

2-2-3　ステータス情報

CF	AHレジスタの値	意　味	FDC STATUSとの対応	
0	0000****(=00H)	Normal end[NT]		
		Ready(SENCEコマンド)[RY]	ST3	D5
0	0001****(=10H)	Control Mark(IMB)[CM]	ST2	D6
		Write Protect(SENCEコマンド)[WP]	ST3	D6
1	0010****(=20H)	DMA Boundary[DB]		
1	0011****(=30H)	End of cylinder[EN]	ST1	D7
1	0100****(=40H)	Equipment check[EC]	ST0	D4
1	0101****(=50H)	Overun[OR]	ST1	D4
1	0110****(=60H)	Not ready[NR]	ST0	D3
1	0111****(=70H)	Not writable[NW]	ST1	D1
1	1000****(=80H)	Error[ER]		
1	1001****(=90H)	Timeout[TO]		
1	1010****(=A0H)	Data error(ID)[DE]	ST1　ST2	D5　D5
1	1011****(=B0H)	Data error(Data)[DD]	ST1　ST2	D5　D5
1	1100****(=C0H)	No data[ND]	ST1	D2
1	1101****(=D0H)	Bad cylinder[BC]	ST2	D1
1	1110****(=E0H)	Missing address mark(ID)[MA]	ST1　ST2	D0　D0
1	1111****(=F0H)	Missing address mark(Data)[MD]	ST1　ST2	D0　D0
0	****0001(=01H)	両面媒体がセットされている	ST3	D3
1	00001***(=08H)	Corrected data[CD]	ハードディスク拡張ステータス	
1	00111***(=78H)	liiegal disk addres[IA]		
1	10001***(=88H)	Direct access an alternate track		
1	10111***(=B8H)	Data error		
1	11001***(=C8H)	Seek error		
1	11011***(=D8H)	代替トラックが読めない		

表2-2-2　ステータス情報一覧

2-2-4　システム共通域

シンボル名・絶対番地	用　途
F2HD_MODE(493H)	7 6 5 4 3 2 1 0 bit0 UNIT #0, bit1 UNIT #1, bit2 UNIT #2, bit3 UNIT #3 — 0：片面モード　1：両面モード bit4 UNIT #0, bit5 UNIT #1, bit6 UNIT #2, bit7 UNIT #3 — 0：48tpiモード(2D／1D)　1：95tpiモード(2DD) 両用タイプFDが1MBモードのときのオペレーションモードの情報を格納します． 初期値は2DD／両面モード(0FFH)です．

DISK_EQUIP2(494H)	7 6 5 4 3 2 1 0 UNIT #0 DA／UA＝F0H UNIT #1 DA／UA＝F1H UNIT #2 DA／UA＝F2H UNIT #3 DA／UA＝F3H 両用タイプインターフェースが640KBモードのとき，接続されているFDDの状態を示します． 1：接続されている 2：接続されていない 両用タイプインターフェースに接続される1MBFDは，インターフェースが640KBインターフェースとなっているとき，DA／UA＝9＊Hではアクセスできません．また，DISK_EQUIPにもセットされません．このドライブはDISK_EQUIP2がセットされ，DA／UA＝F＊Hであるときのみ，アクセス可能です．
DISK_EQUIP (55CH・55DH)	7 6 5 4 3 2 1 0 Y Y Y Y X X X X 3 2 1 0 3 2 1 0 320KBFD UNIT # 1MBFD UNIT #
	15 14 13 12 11 10 9 8 W W W W O O Z Z 3 2 1 0 1 0 640KBFD UNIT # ハードディスク UNIT # 各装置タイプ毎のINITIALIZEを行うと，接続ユニットの状態がセットされます．セットされるFD装置は両用タイプの場合がありSENCEコマンドで確認する必要があります．

表 2-2-3 システム共通域一覧

2-2-5 1MB FDD

データ読み出し(READ DATA)

機能

レジスタALに指定されたデバイスタイプ／ユニットナンバーを持つデバイスの，指定されたID情報を持つセクタ(レジスタCH，CL，DH，DLにより指定)からレジスタES，BPに指定されたメモリ領域(データバッファ領域の先頭アドレス)へ，レジスタBXに指定された長さのレコードを読み込みます．これには，現在のシリンダ位置から読み出す場合と，シークを行ってトラックを選択してから読み出す場合と2つあり，どちらを選択することもできます．READ DATAでは，シングル／マルチトラックの選択(MT)，単密度(FMモード)／倍密度(MFMモード)の選択指定(MF)が可能であり，エラー発生時に8回のリトライを行うか否かをrビットのフラグで指定できます(rビットが0のとき，8回リトライします)．

入　力

割り込みコード：1 BH

AH：BIOS コマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	seek	0	1	1	0

MT……0：シングルトラックリード

　　1：マルチトラックリード(但し，同一シリンダの両面トラックの時のみ)

MF……0：単密度モード

　　1：倍密度モード

r……… 0：8回リトライする

　　1：リトライしない

SEEK……0：シークしない

　　1：シーク動作をする

AL：デバイスタイプ・ユニットナンバー(DA／UA)

7	6	5	4	3	2	1	0
1	0	0	1	0	0	x 2	x 1

(X 2，X 1はドライブに対応……00：0，11：3)

BX：データ長(DTL，バイト単位)

CL：シリンダ番号(C，0～76)

DH：ヘッド番号(H，0～1)

DL：セクタ番号(R，1～26)

CH：セクタ長(N,，00 H～03 H)

CH…………00H　01H　02H　03H

セクタ長……128　256　512　1024(バイト／セクタ)

ES／BP：データバッファアドレスの先頭アドレス

出　力

CF：終了条件(0／1)

AH：ステータス情報

AHレジスタの値	呼　称	内　容
0001****(=10H)	Control mark[CM]	DDAMを検出し，そのセクタを読みだした後正常終了します．
0010****(=20H)	DMA Boundary[DB]	メモリアドレスがバンクにまたがるか，奇数番地から始まるように設定した場合の異常終了ステータスです．
0011****(=30H)	End of cylinder[EN]	1回の動作の転送容量を越えたデータ長(DTL)を指定した場合の異常終了ステータスです．
0100****(=40H)	Equipment check[EC]	デバイスが異常か，RECALIBRATEコマンド実行時，一定時間内にトラック0信号を確認できなかった場合の異常終了ステータスです．
0101****(=50H)	Overun[OR]	セクタ内のデータをメモリに転送している時，一定時間内にデータ転送が終了できなかった場合の異常終了ステータスです．
0110****(=60H)	Not ready[NR]	ユニットの準備ができていない場合の異常終了ステータスです．
1010****(=A0H)	Data error[DE]	ID読み出し時に，CRCエラーが発生した場合の異常終了ステータスです．
1100****(=C0H)	No data[ND]	トラック内に指定されたセクタが見つからなかった場合の異常終了ステータスです．
1110****(=E0H)	Missing address mark mark[MA]	トラック内に指定されたセクタが見つからず，かつIDが一個もなかった場合の異常終了ステータスです．

表2-2-4　AH レジスタのステータス情報(1)

システム共通領域(DISK_RESULT　564H～583H の8バイトエントリ)にはFDCからのリザルトステータス情報を返します．

処　理

・DA／UA に対応する DISK_RESULT の内容を調べます．
・No.4ビット(SEEK)の ON／OFF を調べます．
・DMA の起動．DMA バウンダリの正当性をチェックした後，不正ならば AH に20H を返して終了し，正当ならば DMA を起動します．
・FDC に対し，READ コマンドを送ります．
・マルチトラック MT，MFM モードは BIOS コマンドに従います．
・DDMA(Deleted Data Address)を検出した場合は，そのセクタを読んだ後，正常終了を返します．
・FDC に対し，READ コマンドのパラメータを送出します(シリンダナンバー，ヘッドナンバー，セクタナンバー，セクタ長)．
・EOT(End of Track)，GPL(GAP3の長さ)，DTL(データ長)は表2-2-5のとおりです．
・DTL は256(FFH)です．

MF ビット	N データ長	密　度 (バイト／セクタ)	EOT 最終セクタ	GPL (GAPレングス)
FMモード(単密度)	0 0 H	128	1 A H	0 7 H
	0 1 H	256	0 F H	0 E H
	0 2 H	512	0 8 H	1 B H
MFMモード(倍密度)	0 1 H	256	1 A H	0 E H
	0 2 H	512	0 F H	1 B H
	0 3 H	1024	0 8 H	3 5 H

表2-2-5　MF／N／密度／EOT／GPL表

・割り込み信号を待ちます．入力処理の終了後，リザルトステータスの読み出し，実行終了セクタのID情報の読み出しを行い，システム共通域のRESULT_DATA, BIOSコマンドの出力条件の設定を行います．

・DMAをクローズします．

備　考

・1回の動作の転送容量はMT(マルチトラック)ビット，MF(倍密度指定)ビット，N(セクタ長)に関係します．

MT	MF	N	転送容量(バイト)		最　終　セ　ク　タ
0	0	0 0 H	1～128×n	n＝1～26	ヘッド0のセクタ26，またはヘッド1のセクタ26
	1	0 1 H	1～256×n		
1	0	0 0 H	1～128×n	n＝1～52	ヘッド1のセクタ26
	1	0 1 H	1～256×n		
0	0	0 1 H	1～256×n	n＝1～15	ヘッド0のセクタ15，またはヘッド1のセクタ15
	1	0 2 H	1～512×n		
1	0	0 1 H	1～256×n	n＝1～30	ヘッド1のセクタ15
	1	0 2 H	1～256×n		
0	0	0 2 H	1～512×n	n＝1～8	ヘッド0のセクタ8，またはヘッド1のセクタ8
	1	0 3 H	1～1024×n		
1	0	0 2 H	1～512×n	n＝1～16	ヘッド1のセクタ8
	1	0 3 H	1～1024×n		

表2-2-6　MT，MF，N関係表

・BIOSコマンド実行を正常に終了したときに，システム共通域はRESULT_DATAに設定され，IDR(C，H，R，N)の状態は次表のようになります．

MT	EOT	最終バイトの転送に関したセクタ	IDRの状態			
			C	H	R	N
0	1AH 0FH 08H	ヘッド0のセクタ1～25 ヘッド0のセクタ1～14 ヘッド0のセクタ1～7	もとのまま	もとのまま	R+1	もとのまま
	1AH 0FH 08H	ヘッド0のセクタ26 ヘッド0のセクタ15 ヘッド0のセクタ8	C+1	もとのまま	R=01	もとのまま
	1AH 0FH 08H	ヘッド1のセクタ1～25 ヘッド1のセクタ1～14 ヘッド1のセクタ1～7	もとのまま	もとのまま	R+1	もとのまま
	1AH 0FH 08H	ヘッド1のセクタ26 ヘッド1のセクタ15 ヘッド1のセクタ8	C+1	もとのまま	R=01	もとのまま
1	1AH 0FH 08H	ヘッド0のセクタ1～25 ヘッド0のセクタ1～14 ヘッド0のセクタ1～7	もとのまま	もとのまま	R+1	もとのまま
	1AH 0FH 08H	ヘッド0のセクタ26 ヘッド0のセクタ15 ヘッド0のセクタ8	もとのまま	下1ビットを反転する	R=01	もとのまま
	1AH 0FH 08H	ヘッド1のセクタ1～25 ヘッド1のセクタ1～14 ヘッド1のセクタ1～7	もとのまま	もとのまま	R+1	もとのまま
	1AH 0FH 08H	ヘッド1のセクタ26 ヘッド1のセクタ15 ヘッド1のセクタ8	C+1	下1ビットを反転する	R=01	もとのまま

表2-2-7 MT，EOT，IDR 関係表

データ書き込み(WRITE DATA)

機 能

指定されたデバイスタイプ／ユニットナンバーを持つデバイスの指定された実行開始セクタ(IDR)へ，指定されたメモリ領域の(データバッファ領域の先頭アドレス)指定された長さ(DTL)のデータを書き込みます．書き込みの場合も，「データ読み出し」と同じ様に現在選択されているシリンダ位置のトラックへ書き込む場合と，シークを行ってトラックを選択してから書き込む場合との2つを選べます．また，単密度モード(FM)と倍密度モード(MFM)との書き込み選択指定も可能です．この場合は，マルチトラック指定は使用できないので注意が必要です．

入 力

内部割り込みコード：1BH

AH：BIOS コマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	0	1	0	1

MT……0：シングルトラックライト

1：マルチトラックライト(両面シリンダの同一トラックのみ)

MF……0：単密度モード

1：倍密度モード

r………0：8回リトライする

1：リトライしない

SEEK……0：シークしない

1：シリンダ番号Cへシークする

AL，BX，CH，CL，DH，DL，ES，BPの扱いはREAD時と同じです．

出　力

CF：終了条件……0：正常終了

1：異常終了

AH：ステータス

AHレジスタの値	呼　　称	内　　容
0010****(=20H)	DMA Boundary [DB]	表2-2-4参照
0011****(=30H)	End of cylinder [EN]	表2-2-4参照
0101****(=50H)	Overun [OR]	セクタのデータ部にデータを転送している時，一定時間内にデータ転送が終了できなかった場合の異常終了ステータスです．データを書き込んだ後，ORとなります．
0110****(=60H)	Not ready [NR]	表2-2-4参照
0111****(=70H)	Not writable [NW]	コマンド実行開始時，WRITE PROTECTED信号がオンの場合の異常終了ステータスです．
1010****(=A0H)	Data error [DE]	1セクタのIDを読み取る毎にCRCをチェックし，いずれかのセクタでCRCエラーが生じた場合の異常終了ステータスです．
1100****(=C0H)	No data [ND]	表2-2-4参照
1110****(=E0H)	Missing address mark mark [MA]	表2-2-4参照

表2-2-8　AHレジスタのステータス情報(2)

システム共通域のDISK_RESULT(564H～583H)の8バイトエントリにはFDCからのリザルトステータスが返されます．

処 理

「データ読み出し」と同じ様な処理を行います．ただし，データ長(DTL)で示すバイト数のデータの書き込みがセクタの途中で終了した場合は，残りバイトには00Hを書き込んで，正常終了します．

シーク動作

機 能

指定されたデバイスタイプ／ユニットナンバー(DA／UA)に対して，指定されたシリンダ物理番号(C)までシークさせます．読み出し(read)／書き込み(write)用のBIOSコマンドはシーク動作を同時に行うことが可能です．BIOSコマンド識別コードのb4ビットのON／OFFでシークするか否かを選択することができます．

入 力

内部割り込みコード：1BH

AH：10H(BIOSコマンド識別コード)

AL：シーク動作を行うデバイスタイプ／ユニットナンバー(DA／UA)

CL：シークを行うシリンダ番号(C)……0～76

出 力

CF：終了条件……0：正常終了

　　　　　　　　1：異常終了

AH：ステータス情報……40H：EC(Fault)

　　　　　　　　　　　60H：NR(Not Ready)

システム共通領域DISK_RESULT(564H～586H)の8バイトエントリには現在のシークアドレス(シリンダ番号)がPCNとして格納されます．

処 理

・FDCにシークコマンド(0FH)を送出します．
・FDCにシークコマンドに対するパラメータを送出します．
・シークを行うユニットナンバー，ヘッドナンバーを渡します．
・シーク動作の終了を待ちます．
・シーク動作が終了すると出力のステータス情報を読み出します．

シリンダ0へのシーク(リキャリブレイト)

機　能

シリンダ物理番号0(デバイスからトラック0信号を検出するまで)へシークします．シーク動作は，シリンダ物理番号0の方向へ1シリンダずつ行い，トラック0を検出するまでシークを続けます．

入　力

内部割り込みコード：1BH

AH：07H(BIOS コマンド識別コード)

AL：デバイスタイプ／ユニットナンバー(DA／UA)

出　力

CF：終了条件(0／1)

AH：ステータス情報……40H：不検出(EC となって異常終了)

60H：デバイス ノット レディ

処　理

・FDC にリキャリブレイトコマンド(07H)を送出します．

・次のシーク動作のために約20ミリ sec 待ちます．

トラックのフォーマット

機　能

指定されたセクタ長(N)，トラック当りのセクタ数(SC)，ギャップ数(GPL)，データパターン(D)に従って，1トラック分のフォーマットを行います．セクタのID 部分に書き込むときは，指定されたデータバッファ領域の内容(C，H，R，N の4バイト×セクタ数)を使用します．従って，セクタシーケンス ID や不良シリンダ ID などに書き込むことも可能です．各セクタのデータ部分には，1バイトのデータ D を N で指定された長さ分だけ，繰り返し書き込みます．

入　力

内部割り込みコード：1BH

AH：BIOS コマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
0	MF	r	SEEK	1	1	0	1

AL：デバイスタイプ／ユニットナンバー(DA／UA)

BX：データ長(DTL)

CH：セクタ長(N)……00，01，02，03

CL：シリンダ番号(C)……0～76(ただし，SEEK＝1の時のみ)

DH：ヘッドナンバー(H)……0～1

DL：データ部への書き込みデータパターン(D)

ES／BP：データバッファ領域の先頭アドレス

出 力

CF：終了条件(0／1)

AH：ステータス情報

AHレジスタの値	呼　称	内　容
0010****(=20H)	DMA Boundary [DB]	表2-2-4参照
0100****(=40H)	Equipment check [EC]	書き込み終了時，Fault 状態を検出した場合の異常終了ステータスです．
0101****(=50H)	Overun [OR]	表2-2-8参照
0110****(=60H)	Not ready [NR]	表2-2-4参照
0111****(=70H)	Not writable [NW]	表2-2-8参照

表2-2-9　AH レジスタのステータス情報(3)

処 理

・DA／UA で指定されるユニットで，現在選択されているシリンダの H バイトで示されるヘッド面にあるトラックにフォーマットを書き込みます．

・セクタ ID 部の書き込みは，指定されているデータバッファ領域の先頭アドレス(ES：BP)から，BX で示されるデータ長までのデータバッファ上に，セクタ ID ごとの C，H，R，N の4バイトエントリをトラックのセクタ数分だけ展開しておき，これを書き込みます．

・セクタのデータ部の書き込みは，指定された DL(データパターン D)の内容を，指定された CH(セクタ長 N)の長さ分だけセクタごとに繰り返します．

備 考

セクタ ID 部をフォーマットするための準備は次のとおりです．

ES：BP	C	論理シリンダ番号
+1	H	論理ヘッド番号
+2	R	論理セクタ番号(セクタシーケンス)
+3	N	物理セクタ長
	：	
+255		

図2-2-2　データバッファの構造　※C，H，R，N とも1セクタに対する ID 部の情報です．

指定方法		指定内容		トラック当りのセクタ数	ギャップ長	IDフォーマット用データバッファ長	備考
MFビット	セクタ長 N	密度	セクタ当りのバイト数	SC	GPL	(DTL)	
0	00	単密度 FMモード	128	1AH (26)	1BH	104	N88-BASIC (シリンダ0, HEAD0) MS-DOS(8″1S)
	01		256	0FH (15)	2AH	60	
	02		512	08H (8)	3AH	32	
1	01	倍密度 MFMモード	256	1AH (26)	36H	104	N88-BASIC (シリンダ0, HEAD0以外)
	02		512	0FH (15)	54H	60	
	03		1024	08H (8)	74H	32	MS-DOS

注：DTLは上表の値以上であればかまいません．255をとるようにします．

表2-2-10　パラメータ指定方法

セクタシーケンス ＼ 物理セクタ番号	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A
00, 01 (CP／M, MS-DOS)	物理セクタ番号と同じです
0D (BASIC)	01 0E 02 0F 03 10 04 11 05 12 06 13 07 14 08 15 09 16 0A 17 0B 18 0C 19 0D 1A

表2-2-11　26セクタ／トラック

セクタシーケンス ＼ 物理セクタ番号	01 02 03 04 05 06 07 08
00, 01 (MS-DOS)	物理セクタ番号と同じです

表2-2-12　8セクタ／トラック

初期化(イニシャライズ)

機　能

1MB フロッピーディスク装置全体の初期化を行います．

・FDC μPD765A の初期化．

・システム共通領域(DISK_EQUIP，DISK_UNIT，DISK_RESULT)の初期化．

・FDCに対し，SPECIFYコマンドを送ります(FDCのモード設定).
・各装置にリキャリブレイトコマンドを送ります(接続状況と動作確認).

入 力

内部割り込みコード：1BH

AH：03H(BIOSコマンド識別コード)

AL：デバイスタイプ／ユニットナンバー(DA／UA)

出 力

CF：終了条件(0／1)

AH：ステータス情報

処 理

・FDC μPD765Aの初期化(ライトコントロールレジスタを使用).
・システム共通領域の初期化.
・FDCに対する初期値の設定(SPECIFYコマンド).

HUT(Head Unload Time)の設定

リード／ライトコマンド実行後，アンロード状態に達するまでの時間を設定します.

SRT(Step Rate Time)の設定

シーク動作時，デバイスへ送るStep信号の間隔時間を指定します.

HLT(Head Load Time)の設定

ヘッドをロードさせた後，安定状態になるまでの待ち時間を指定します.

DMAモードの設定

・各ユニットに対して，リキャブレイトコマンドによるチェックを行います.

ベリファイ

機 能

指定されているデバイスタイプ／ユニットナンバー(DA／UA)のディスク装置の，指定されたセクタのデータを読み取ります．ただし，メモリへは転送しません．指定条件は「データ読み出し」とほとんど同じです.

入 力

内部割り込みコード：1BH

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	0	0	0	1

出　力

DDAM(Deleted Data Address Mark)を検出してもスキップして，処理を続行します(他は「データ読み出し」を参照してください).

処　理

読み取ったデータをメモリに転送しないことを除いて，「データ読み出し」の場合と同じです.

センス

機　能

指定したデバイスの状態を調べます.

入　力

内部割り込みコード：1BH

AH：04H(BIOSコマンド識別コード)

AL：デバイスタイプ／ユニットナンバー(DA／UA)

出　力

CF：終了条件(0／1)

AH：ステータス情報……10H：WP(ライトプロテクト)

60H：NR(ノットレディ)

ビット0：1(両面媒体)

ビット0：0(片面媒体)

処　理

FDCへSENSEコマンドを送り，デバイスの状態を取得します．これを編集して，AHにセットします.

IDの読み出し(READ ID)

機　能

指定されたデバイスタイプ／ユニットナンバーのディスク装置の指定されたトラックのIDを読み取り，IDR(C, H, R, N)に格納します．

入　力

内部割り込みコード：1BH

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
0	MF	r	SEEK	1	0	1	0

AL：デバイスタイプ／ユニットナンバー

CL：シリンダ番号(SEEK ONの時のみ意味を持ちます)

DH：ヘッド番号

出　力

CF：終了条件(0／1)

AH：ステータス情報

AHの内容	略　称	ステータス呼称	内　容
40H	EC	Equipment Check	「データ読み出し(IMB)」参照
60H	NR	Not Ready	「データ読み出し(IMB)」参照
C0H	ND	No Data	1トラック分の読み取りを行っても，正しいIDがみつからない状態で異常終了．
E0H	MA	Missing Address Mark	正しいIDがみつからず，しかもIDアドレスマークが1回も検出されなかった．

表2-2-13　AHレジスタのステータス情報(4)

CH：セクタ長

CL：シリンダ番号

DH：ヘッド番号

DL：セクタ番号

処　理

指定されたトラックで，最初に正常に読み取れたIDをIDRに格納します．

デッドリーデータの書き込み

機　能

セクタのセンターフィールドのData Address Markの代わりにDeleted Data Address Markを書き込むことを除いて，「データ書き込み」の場合と同じです．

入　力

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	1	0	0	1

他はWrite Dataコマンドと同じです．

出　力

「データ書き込み」と同じです．

処　理

「データ書き込み」と同じです．

デッドリーデータの読み出し

機　能

セクタのセンターフィールドにあるData Address Markの代わりにDeleted Data Address Markを扱う事を除いて，「データ読み出し」の場合と同じです．

入　力

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	1	1	0	0

他は「データ読み出し」と同じです．

出 力

「データ読み出し」と同じです.

処 理

「データ読み出し」と同じです.

診断のための読み出し(Read Diagnostic)

機 能

インデックスマークの直後から読み取りを開始し，IDエラー，及びデータ部のエラーが検出されても読み取りを続行することを除いて，「データ読み出し」と同じです.

入 力

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
0	MF	r	SEEK	0	0	1	0

他は「データ読み出し」と同じです.

出 力

「データ読み出し」と同じです.

処 理

インデックスマークの直後から読み取りを開始します. 次の3点を除いて,「データ読み出し」と同じです.

- IDまたはデータ部のCRCエラーが検出されても読み取りを続けます.
- 読み取ったIDとIDRの比較を行います. その結果が等しくなくても，ステータスにND(No Data)をセットするだけで，そのセクタのデータを処理します.
- Deleted Data Address Markのあるセクタを検出しても，コマンドの終了条件にはなりません.

2-2-6　640KB FDD

データ読み出し(READ DATA)

機　能

1MB FDDの場合とほとんど同じです．デバイスタイプ／ユニットナンバーの指定(DA／UA)が70H～73Hとなるだけです．

入　力

AHに入力する8ビットは1MB FDDの場合と同じです．

AL：以下の8ビットを入力(70H～73H)

7	6	5	4	3	2	1	0
0	1	1	1	0	0	X2	X1

(X2，X1は接続ディスクドライブに対応……00：0／11：3)

DLにセットするセクタ番号が0～16になる以外，IDRの扱いは1MB FDDと同じです．

出　力

CF／AHに返ってくるものは1MB FDDと同じです．システム共通領域(F2DD_RESULT　5D0H～5DFHの16バイトエントリ)にはFDCからのリザルトステータスを格納します．

処　理

・システム共通領域(F2DD_RESULT)の内容を調べます．

・SEEKビットのON／OFFを判定し，シークが終了したか，またはシークビットがOFFならば，3の動作を行います．

・DMAの起動(1MB FDDと同じです)．

・FDCにREADコマンドを送ります(1MB FDDと同じです)．

・FDCに対し，READコマンドのパラメータ(C，H，R，N)を送出します．

・EOT，GPL，DTLは次表の通りです．

・DTLは255(FEH)です．

・割り込み信号の入力を待ちます(1MB FDDの場合と同じです)．

・DMAをクローズします．

MF ビット	セクタ内の データ長N	セクタ当りの密度 (バイト／セクタ)	EOT (トラック上の最終セクタ)	GPL (GAPレングス)
FMモード (単密度)	00H	128	10H	07H
	01H	256	09H	0EH
	02H	512	05H	1BH
MFMモード (倍密度)	01H	256	10H	0EH
	02H	512	09H	1BH
	03H	1024	05H	35H

表2-2-14　EOT，GPL 対応表

備　考

・転送容量と MT，MF，N ビットの関係は次表のようになります．

MT	MF	N	転送容量(バイト)		最　終　セ　ク　タ
0	0	00H	1～128×n	n=1～16	ヘッド0のセクタ16またはヘッド1のセクタ16
	1	01H	1～256×n		
1	0	00H	1～128×n	n=1～32	ヘッド1のセクタ16
	1	01H	1～256×n		
0	0	01H	1～256×n	n=1～9	ヘッド0のセクタ9またはヘッド1のセクタ9
	1	02H	1～512×n		
1	0	01H	1～256×n	n=1～18	ヘッド1のセクタ9
	1	02H	1～512×n		
0	0	02H	1～512×n	n=1～5	ヘッド0のセクタ5またはヘッド1のセクタ5
	1	03H	1～1024×n		
1	0	02H	1～512×n	n=1～10	ヘッド1のセクタ5
	1	03H	1～1024×n		

注：nの値はプログラマブルで，EOT の値で定義されます．

表2-2-15　MT，MF，N 対応表

・BIOS コマンドを実行したときの RESULT_DATA に設定されます．IDR(C, H, R, N)の状態は次表のようになります．

MT	EOT	最終バイトの転送に関係したセクタ	IDRの状態			
			C	H	R	N
0	10H 09H 05H	ヘッド0のセクタ1～15 〃　1～8 〃　1～4	もとのまま	もとのまま	R+1	もとのまま
	10H 09H 05H	ヘッド0のセクタ16 〃　9 〃　5	C+1	もとのまま	R=01(注)	もとのまま
	10H 09H 05H	ヘッド1のセクタ1～15 〃　1～8 〃　1～4	もとのまま	もとのまま	R+1	もとのまま
	10H 09H 05H	ヘッド1のセクタ16 〃　9 〃　5	C+1	もとのまま	R=01(注)	もとのまま
1	10H 09H 05H	ヘッド0のセクタ1～15 〃　1～8 〃　1～4	もとのまま	もとのまま	R+1	もとのまま
	10H 09H 05H	ヘッド0のセクタ16 〃　9 〃　5	もとのまま	下1ビットを反転する	R=01(注)	もとのまま
	10H 09H 05H	ヘッド1のセクタ1～15 〃　1～8 〃　1～4	もとのまま	もとのまま	R+1	もとのまま
	10H 09H 05H	ヘッド1のセクタ16 〃　9 〃　5	C+1	下1ビットを反転する	R=01(注)	もとのまま

注：ヘッドの選択状態(ST0のHDビットも含む)は最終バイトの転送に関係したセクタに属するトラックのままです．

表2-2-16　MT，IDR対応表

データ書き込み(WRITE DATA)

機　能

デバイスタイプ／ユニットナンバーの指定が70H～73Hに変わるだけで，後は1MB FDDの場合と同じです．

入　力

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	0	1	0	1

AL，BX，CH，CL，DH，DL，ES／BP に渡すパラメータも READ の場合と同じように指定します．

出 力

CF，AH に返ってくる値は1MB FDD の場合と同じです．

AH の内容	略 称	ステータス呼称	内 容
20H	DB	DMA Boundary	「6.3.1 データ読み出し(640K)」参照
30H	EN	End of Cylinder	「6.3.1 データ読み出し(640K)」参照
40H	EC	Equipment Check	データの書き込み時，デバイスのFAULT状態を検出した場合の異常終了ステータス．
50H	OR	Over Run	セクタのデータ部にメモリからデータ転送をしているとき，一定時間内にデータを転送できなかった場合の異常終了ステータス．そのセクタを書き込み後，ORとなる．
60H	NR	Not Ready	「6.3.1 データ読み出し(640K)」参照
70H	NW	Not Writable	コマンド実行開始時，WRITE PROTECTED信号がオンの場合の異常終了ステータス．
A0H	DE	Data Error	1セクタのIDを読み取るごとにCRCバイトをチェックし，いずれかのセクタでCRCエラーが生じた場合の異常終了ステータス．
C0H	ND	No Data	「6.3.1 データ読み出し(640K)」参照
E0H	MA	Missing Address mark	「6.3.1 データ読み出し(640K)」参照

表2-2-17 AH レジスタのステータス情報(5)

システム共通領域(F2DD_RESULT の5D0H～5DFH の16バイトエントリ)には FDC からのリザルトステータスが格納されます．

処 理

「データ読み出し」と同様の処理を行います．DTL で示されるバイト数のデータの書き込みがセクタの途中で終了した場合，残りのバイトには00H を書き込んで終了します．

シーク動作

機 能

1MB FDD の場合と同じです．

入 力

以下の点を除いて，1MB FDD の場合と同じです．

AH：BIOS コマンド識別コード(10H)

AL ：デバイスタイプ／ユニットナンバー(70H～73H)

CL ：シークを行うシリンダ番号(0～76)

出　力

以下の点を除いて，1MB FDD の場合と同じです.

CF：終了条件(0／1)

AH：ステータス情報

システム共通領域(F2DD_RESULT)：現在のシークアドレス(シリンダ番号)

処　理

1MB FDD の場合と同じです.

シリンダ0へのシーク(リキャリブレイト)

機　能

1MB FDD の場合と同じです.

入　力

AH：以下の8ビットを入力

7	6	5	4	3	2	1	0
0	0	r	0	0	1	1	1

AL：デバイスタイプ／ユニットナンバー(70H～73H)

出　力

1MB FDD の場合と同じです.

処　理

FDC は0シリンダに向かって77回シークを行いますが，シリンダ数が80なので FDC に対して，2回のリキャリブレイトコマンドを実行します.

トラックのフォーマット

機　能

1MB FDD の場合と同じです.

入 力

AH：BIOS コマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
0	MF	r	SEEK	1	1	0	1

AL：DA／UA(70H～73H)

BX, CH, CL, DH, DL, ES／BP は1MB FDD の場合と同じです.

出 力

1MB FDD の場合と同じです.

処 理

1MB FDD の場合と同じです.

備 考

セクタのID部をフォーマットするための準備は次のとおりです.

指定方法		指定内容		トラック当りのセクタ数	ギャップ長	IDフォーマット用データバッファ長	備考
MFビット	セクタ長 N	密度	セクタ当りのバイト数	SC	GPL	(DTL)	
0	00	単密度	128	10H (16)	1BH	64	
	01	FM モード	256	09H (9)	2AH	36	
	02		512	05H (5)	3AH	20	
1	01	倍密度	256	10H (16)	33H	64	N88-BASIC
	02	MFM モード	512	09H (9)	50H	36	IBMの8セクタ／トラックとするときは32バイトにする
	03		1024	05H (5)	74H	20	

注：DTL は上表の値以上であればかまいません．255をとるようにします．

表2-2-18 パラメータの指定方法

・データバッファの内容は1MB FDD と同様

・セクタシーケンスに従った論理セクタ番号

セクタシーケンス＼物理セクタ番号	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
00,01 (CP／M, MS-DOS)	物理セクタ番号と同じです
08 (BASIC)	01 09 02 0A 03 0B 04 0C 05 0D 06 0E 07 0F 08 10

表2-2-19　16セクタ／トラック表

セクタシーケンス＼物理セクタ番号	01 02 03 04 05 06 07 08
00,01 (MS-DOS)	物理セクタ番号と同じです

表2-2-20　8セクタ／トラック表

セクタシーケンス＼物理セクタ番号	01 02 03 04 05 06 07 08 09
00,01 (MS-DOS)	物理セクタ番号と同じです

表2-2-21　9セクタ／トラック表

初期化(イニシャライズ)

機　能

640KB FDD装置全体の初期化を行います．

・μPD765Aの初期化．
・システム共通領域(DISK_EQUIP，DISK_INT，DISK_RESULT)の初期化．
・FDCに対し，SPECIFYコマンドを送ります．
・各装置にリキャリブレイトコマンドを送ります．

入　力

内部割り込みコード：1BH

AH：03H

AL：デバイスタイプ／ユニットナンバー(70H～73H)

出　力

1MB FDDの場合と同じです．

処　理

1MB FDDの場合と同じです．

ベリファイ

機　能

1MB FDDの場合と同じです.

入　力

AH：以下の8ビットを入力

7	6	5	4	3	2	1	0
MT	MF	r	SEEK	0	0	0	1

他は「データ読み出し」と同じです.

出　力

1MB FDDの初期化と同じです.

処　理

1MB FDDの初期化と同じです.

センス

機　能

デバイスの状態を調べます.

入　力

AH：04H

AL：DA／UA(70H～73H)

出　力

CF：終了条件(0／1)

AH：ステータス情報

処　理

1MB FDDのセンスの場合と同じです.

AHの内容	略　称	ステータス呼称	内　　　容
10H	WP	Write Protect	媒体がセットされているが，ライトプロテクトの状態である．
60H	NR	Not Ready	媒体がセットされていない．
ビット0			0：片面モード　（注） 1：両面モード
ビット2			0：40シリンダモード　（注） 1：80シリンダモード

注：Set Operation Mode コマンドで設定した値が UNIT 毎に通知されます．

表2-2-22　AH レジスタのステータス情報(6)

IDの読み出し(READ ID)

機　能

1MB FDDの「IDの読み出し」と同じです．

入　力

AH：以下の8ビットを入力

7	6	5	4	3	2	1	0
0	MF	r	SEEK	1	0	1	0

AL：DA／UA(70H～73H)

CL：シリンダ番号(ただし，SEEKがONの時のみ)

CH：ヘッド番号(0～1)

出　力

1MB FDDの場合と同じです．

処　理

1MB FDDの場合と同じです．

2-2-7 1MB／640KB 両用 FDD

1MB／640KB 両用タイプインターフェースは，1MB モードと640KB モードの2つをサポートしています(PC-9801VM／UV のみ)．

このインターフェースは，640KB モードでは PC-9801／E／F／M／U2／VF における640KB インターフェースと同様な機能を実現するもので，デバイスタイプとして70H～73H を使用します．

また，1MB インターフェースモードでは，PC-9801／E／F／M／U2／VF における1MB インターフェースの機能を含んだ上に，640KBFD をアクセスする機能も持っています．このモードの場合は，デバイスタイプとして次のような値を使用します．

- 1MB FD アクセス………90H-93H
- 640KB FD アクセス…… 10H-13H

1MB／640KB 両用インターフェースはリセット後，ディップスイッチの設定によってどちらかのモードに設定されます．

両用タイプでは1MB／640KB どちらかのモードに応じて既に述べたコマンドが使用可能です．このページでは両用タイプに特有なコマンドだけを述べることにします．

新センス(COMMAND／STATUS)

機能

指定したデバイスの状態を調べます．

入力

内部割り込みコード：1BH

AH：84H

AL：DA／UA(70H～73H，90H～93H，10H～13H)

出力

CF：終了条件(0／1)

AH：ステータス情報

7	6	5	4	3	2	1	0
×	×	×	×	0	0	0	0

×……「センス」と同じ
①……0：片面媒体
　　　1：両面媒体(1MB／640K両用ドライブでは常に両面)
②……0：1MBドライブ
　　　1：1MB／640KB両用ドライブ

図2-2-13 1MB ステータス(1MB インターフェースモード)

7	6	5	4	3	2	1	0
×	×	×	×	④	③	②	①

×……「センス」と同じ
①……0：片面モード
　　　1：両面モード
②……0：AI有り（1MBドライブ時は常に0）
　　　1：AI無し
③……0：40シリンダモード（1MBドライブ時は常に0）
　　　1：80シリンダモード
④……0：640KBドライブ
　　　1：1MB／640KB両用ドライブ
※エラー発生時は，ビット3～0のうちビット3のみ有効
※③は640KBモードの時のみ有効

図2-2-14　640KBステータス（640KBインターフェースモード）

SET OPERATION MODE（1MBモード時のみ）

機　能

両用ドライブを1MBインターフェースモードで使用する際，640KB FDをアクセスする場合のモードを指定します．

入　力

内部割り込みコード：1BH

AH：0EHまたは8EH

AL：以下の8ビット

7	6	5	4	3	2	1	0
0	0	0	1	U3	U2	U1	U0

U0～U3：ユニットナンバー

AH：0EHをセットしたとき，各ユニットのビットは
　0のとき：片面モード
　1のとき：両面モード

AH：8EHをセットしたとき，各ユニットのビットは
　0のとき：48tpiモード
　1のとき：96tpiモード

（各ビットの初期値はすべて1です）

出 力

CF：0かつ AH：0……正常終了

CF：1かつ AH：40H……異常終了

新センスコマンドで確認できます(PC-9801E／F／M では AH：40H，CF：1)．

新イニシャライズ(640KB インターフェースモード時のみ)

機 能

640KB インターフェースモード時，AI(Attemtion Intruppt)を検出するように初期化します．AI なしに戻すことはできません．

入 力

内部割り込みコード：1BH

AH：83H

AL：7xH または FxH(x はユニットナンバー)

出 力

CF：0かつ AH：0……正常終了

CF：1かつ AH：40H……異常終了

AI が有効か否かは，このコマンドまたは新センスコマンドで確認します(PC-9801E／F／M では AL：7xH の場合正常終了となるので，AL：FxH として実行すること)．

2-2-8 ハードディスク

●内部割り込みコード：1BH(N88-BASIC(86)の場合，0B1H でも可)

レジスタ名	AH	AL	BX	CX	DH	DL	ES	BP
レジスタの用途 ＼ BIOS コマンド名	コマンド識別コード(注1)	DA／UA (80H-81H)	データサイズ 256×n バイト	シリンダ番号 5MB 0～152 10MB 0～309 20MB 0～307	ヘッド番号 5，10MB (0～3) 20MB (0～7)	セクタ番号 (0～32)	データバッファ先頭アドレスセグメントベース	データバッファ先頭アドレスオフセット
READ DATA	α6H	○	○	○	○	○	○	○
WRITE DATA	α5H	○	○	○	○	○	○	○
RECALI-BRATE	α7H	○						
RETRACT	αFH	○						

FORMAT TRACK／DRIVE	βDH	○	BH インタリーブファクタ (1～16)	○	○	○		
INITIALIZE	03H	DA (8×H)						
VERIFY	αIH	○	○	○	○	○	○	○
SENSE	04H	○						
ASSIGN ALTERNATE TRACK	08H	○	○	○	○	○	○(注2)	○(注2)
FORMAT BAD TRACK	0BH	○	○	○	○	0 (ゼロ)	○(注2)	○(注2)

注1：AHの上位4ビットα($b_7b_6b_5b_4$)＝××$\overline{r}$×　$\overline{r}$：リトライ指定ビット
　　 AHの上位4ビットβ($b_7b_6b_5b_4$)＝d×$\overline{r}$×　d：フォーマット単位指定ビット
注2：代替トラックアドレスを読み込む4バイトバッファを指定します．

表2-2-23　入力データ一覧

CF (注1)	AH		説明	
	16進表示	ビット(注2)	略称	内容
0	00H	0 0 0 0××××	NT	Normal end
0	00H		RY	Ready (Sense コマンド)
0	10H	0 0 0 1××××	CM	Control Mark
0	10H		WP	Write Protect (Sense コマンド)
1	20H	0 0 1 0××××	DB	DMA Boundary
1	30H	0 0 1 1××××	EN	ENd of cylinder
1	40H	0 1 0 0××××	EC	Equipment Check
1	50H	0 1 0 1××××	OR	OverRun
1	60H	0 1 1 0××××	NR	Not Ready
1	70H	0 1 1 1××××	NW	Not Writable
1	80H	1 0 0 0××××	ER	ERror
1	90H	1 0 0 1××××	TO	TimeOut
1	A0H	1 0 1 0××××	DE	DataError (ID)
1	B0H	1 0 1 1××××	DD	DataError (Data)
1	C0H	1 1 0 0××××	ND	No Data
1	D0H	1 1 0 1××××	BC	Bad Cylinder
1	E0H	1 1 1 0××××	MA	Missing Address mark (ID)
1	F0H	1 1 1 1××××	MD	Missing address mark (Data)
1	08H	0 0 0 0 1×××	CD	Corrected Data
1	78H	0 0 1 1 1×××	IA	Illegal disk Address
1	88H	1 0 0 0 1×××		Direct access an alternate track
1	B8H	1 0 1 1 1×××		Data Error
1	C8H	1 1 0 0 1×××		Seek error
1	D8H	1 1 0 1 1×××		代替トラックが読めない

注1：CF＝0：正常終了，CF＝1：異常終了
注2：AHレジスタの内容で×になっているビット値は無視する(チェックする必要がない)．16進表示ではゼロとみています．

表2-2-24　ステータス一覧

エラーステータス情報			リトライ処理
略　称	内　　　容	AHの内容	
DB EN	DMA Boundary ENd of cylinder	0 0 1 0×××× 0 0 1 1××××	コマンド使用上に誤りがある
EC OR NR NW	Equipment Check Over Run Not Ready Not Writable	0 1 0 0×××× 0 1 0 1×××× 0 1 1 0×××× 0 1 1 1××××	再試行する
DE	Data Error	1 0 1 0××××	Recalibrate→Seek→リード／ライト系コマンド
ND MA	No Data Missing Address mark	1 1 0 0×××× 1 1 1 0××××	

表2-2-25　エラーリトライ処理

●相対アドレスによるアクセス

ハードディスクにアクセスする場合，ボリューム上の相対アドレスによってアクセスすることができます．以下に使い方を示します．

①デバイスタイプ／ユニットナンバー(DA／UA)には00H～01Hをセットします．

②相対アドレスをセットします．

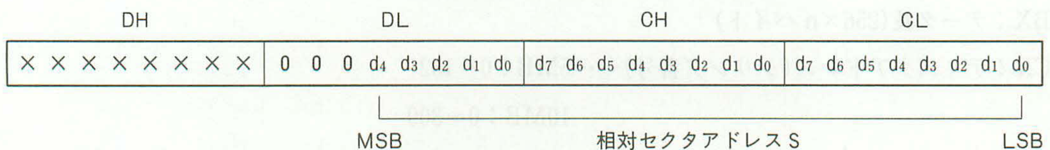


図2-2-25　相対アドレスの指定

③相対セクタアドレスと絶対セクタアドレスとの関係は次のとおりです．

- 相対セクタアドレスS＝33(4X3＋X2)＋X1
- 絶対セクタアドレス(X3，X2，X1)
- シリンダ番号 X3：(0～152または0～309または0～307)
- ヘッド番号　X2：(0～3または0～7)
- セクタ番号　X1：(0～32)

④IDRを指定するコマンドに適用されます．

データ読み出し(READ DATA)

機　能

指定されたデバイスの指定されたディスクアドレスから，指定された長さのデータを，メモリ上の指定されたデータバッファへ読み出します．

入　力

内部割り込みコード：0B1H

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
*	*	r	*	0	1	1	0

(*は任意のビット)

AL：デバイスタイプ・ユニットナンバー(80H～81H)

BX：データ長(256×nバイト)

CX：ディスクアドレス(シリンダ番号)……5MB：0～152
　10MB：0～309
　20MB：0～307

DH：ディスクアドレス(ヘッド番号)……5,10MB：0～3
　20MB：0～7

DL：ディスクアドレス(セクタ番号)……0～32

ES／BP：データバッファ先頭アドレス

出　力

CF：終了条件(0／1)

AH：ステータス情報

処　理

ALで指定されたデバイスタイプ／ユニットナンバー(DA／UA)のデバイス上の，指定されたディスクアドレスIDR(CX，DH，DL)から，指定された長さ(BX)のデータを受け取り，主記憶装置上の指定されたデータバッファに(先頭アドレスES：BPからBXで指定された長さ)へ転送します．ハードウェア上の動作は次のようになります．

・1セクタ分のデータ転送後，指定されたセクタ分のデータが転送されていなければ，次のセクタを処理します．

・指定されたトラックのデータが，トラック，またはシリンダにまたがっていた場合，自動的にトラックまたはシリンダを切り替えて処理を続けます．

・指定されたセクタ数だけのデータを転送したら，正常終了します．コマンド実行中にエラーが発生した場合は，その内容をAHにステータスとしてセットし，異常終了します．ただし，rビットがONの時は，8回リトライをします．

データ書き込み(WRITE DATA)

機　能

メモリ上の指定されたデータバッファから，指定されたデバイスの指定されたディスクアドレスへ指定された長さのデータを書き込みます．

入　力

内部割り込みコード：0B1H

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
*	*	r	*	0	1	0	1

(＊は任意のビット)

AL，BX，CX，DH，DL，ES／BPの指定は「データ読み出し」と同様です．

出　力

CF：終了条件(0／1)

AH：ステータス情報

処　理

メモリ上の指定されたデータバッファの先頭アドレス(ES：BP)から，指定された長さの(BX)(256×nバイト)のデータを，ALで指定したデバイスのディスクアドレスIDR(CX, DH, DL)へ書き込みます．ハードウェア上の動作，エラー検出時の動作は「データ読み出し」コマンドと同じです．

シリンダ0へのシーク(リキャリブレイト)

機 能

指定されたデバイスのアームをシリンダ0へシークさせます.

入 力

割り込みコード：0B1H

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
*	*	r	*	0	1	1	1

AL：DA／UA

出 力

CF ：終了条件(0／1)

AH：ステータス情報

処 理

AHで指定されたデバイスのアームを物理シリンダ0へ移動します.エラーリトライ動作は,「データ読み出し」コマンドと同じです.

リトラクト

機 能

デバイスタイプ／ユニットナンバー(DA／UA)で指定されるデバイスに対して，アームを不使用シリンダへ移動させます.

入 力

内部割り込みコード：0B1H

AH：以下の8ビットを入力

7	6	5	4	3	2	1	0
*	*	r	*	1	1	1	1

AL：DA／UA(80H-81H)

出　力

CF：終了条件(0／1)

AH：ステータス情報

処　理

DA／UAで指定されるデバイスのアームを，意味のないシリンダ位置(データを書き込まない不使用シリンダ)に移動させます．コマンド実行中にエラーを検出した時には，そのエラーステータスをAHにセットし，異常終了します．ただし，rのビットがONの時はエラー通知の前に8回のリトライをします．リトライによって，成功すれば，エラーを通知せずに正常終了します．

注　意

ハードディスクの電源を切る前には必ずリトラクトコマンドを送り，アームを不使用シリンダに移動させなければなりません．

IDの書き込み(FORMAT TRACK／DRIVE)

機　能

デバイスタイプ／ユニットナンバー(DA／UA)で指定されるデバイスに対して，トラック単位またはデバイス単位にセクタフォーマットを行います．ID部にはインタリーブファクタに従ったセクタ番号が書き込まれ，データ部にはE5Hが書き込まれます．

入　力

内部割り込みコード：0B1H

AH：以下の8ビットを入力

7	6	5	4	3	2	1	0
d	*	r	*	1	1	0	1

(dはフォーマット単位の指定……0ならばトラック単位)

AL：デバイスタイプ／ユニットナンバー(DA／UA)

BX：インタリーブファクタ(1～16)

CX：トラック単位にフォーマットする場合のシリンダ番号(デバイス単位の場合は0)

DH：トラック単位にフォーマットする場合のヘッド番号(デバイス単位の場合は0)

DL：0

出　力

CF：終了条件(0／1)

AH：ステータス情報

処　理

正常終了の時は機能の通り，指定されたフォーマットを行います．エラーを検出したときには，その内容をAHに格納し，異常終了します．ただし，リトライビットがONの時は，エラー通知の前に8回のリトライをし，エラーの種類に応じてリキャリブレイト，再シークが行われます．再試行が成功すれば，エラー通知をせずに，正常終了します．

＜インタリーブファクタについて＞

指定されたフォーマット単位によって，それぞれのトラックをセクタごとにフォーマットします．ID部のセクタアドレス(シリンダ番号，ヘッド番号，セクタ番号)にはインタリーブファクタに従った論理セクタ番号が書き込まれます．物理的なセクタシーケンスとID部に書き込まれる論理セクタ番号は次表のようになります．

インタリーブファクタ(16進)	物理セクタ番号　(16進)																															
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
	論理セクタ番号　(16進)																																
01	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
02	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E	20	01	03	05	07	09	0B	0D	0F	11	13	15	17	19	1B	1D	1F
03	00	03	06	09	0C	0F	12	15	18	1B	1E	01	04	07	0A	0D	10	13	16	19	1C	1F	02	05	08	0B	0E	11	14	17	1A	1D	20
04	00	04	08	0C	10	14	18	1C	20	01	05	09	0D	11	15	19	1D	02	06	0A	0E	12	16	1A	1E	03	07	0B	0F	13	17	1B	1F
05	00	05	0A	0F	14	19	1E	01	06	0B	10	15	1A	1F	02	07	0C	11	16	1B	20	03	08	0D	12	17	1C	04	09	0E	13	18	1D
06	00	06	0C	12	18	1E	01	07	0D	13	19	1F	02	08	0E	14	1A	20	03	09	0F	15	1B	04	0A	10	16	1C	05	0B	11	17	1D
07	00	07	0E	15	1C	01	08	0F	16	1D	02	09	10	17	1E	03	0A	11	18	1F	04	0B	12	19	20	05	0C	13	1A	06	0D	14	1B
08	00	08	10	18	20	01	09	11	19	02	0A	12	1A	03	0B	13	1B	04	0C	14	1C	05	0D	15	1D	06	0E	16	1E	07	0F	17	1F
09	00	09	12	1B	01	0A	13	1C	02	0B	14	1D	03	0C	15	1E	04	0D	16	1E	05	0E	17	20	06	0F	18	07	10	19	08	11	1A
0A	00	0A	14	1E	01	0B	15	1F	02	0C	16	20	03	0D	17	04	0E	18	05	0F	19	06	10	1A	07	11	1B	08	12	1C	09	13	1D
0B	00	0B	16	01	0C	17	02	0D	18	03	0E	19	04	0F	1A	05	10	1B	06	11	1C	07	12	1D	08	13	1E	09	14	1F	0A	15	20
0C	00	0C	18	01	0D	19	02	0E	1A	03	0F	1B	04	10	1C	05	11	1D	06	12	1E	07	13	1F	08	14	20	09	15	0A	16	0B	17
0D	00	0D	1A	01	0E	1B	02	0F	1C	03	10	1D	04	11	1E	05	12	1F	06	13	20	07	14	08	15	09	16	0A	17	0B	18	0C	19
0E	00	0E	1C	01	0F	1D	02	10	1E	03	11	1F	04	12	20	05	13	06	14	07	15	08	16	09	17	0A	18	0B	19	0C	1A	0D	1B
0F	00	0F	1E	01	10	1F	02	11	20	03	12	04	13	05	14	06	15	07	16	08	17	09	18	0A	19	0B	1A	0C	1B	0D	1C	0E	1D
10	00	10	20	01	11	02	12	03	13	04	14	05	15	06	16	07	17	08	18	09	19	0A	1A	0B	1B	0C	1C	0D	1D	0E	1E	0F	1F

表2-2-26　インタリーブファクタ表

注 意

・同一ボリューム上に異なるインタリーブファクタのトラックを持つことはできません．コントローラは最初のアクセスで，インタリーブファクタを記憶します．

・出荷時の不良トラックは，不良トラックフラグを立てて，不良トラック扱いにしています．ゆえに，フォーマットを行うときは，このトラックの先頭セクタを読み，不良トラックでないことを確認してから，フォーマットしなければなりません．

初期化

機 能

接続されているハードディスクユニット・インターフェースおよびBIOS情報の初期設定を行います．

入 力

内部割り込みコード：01BH

AH：BIOSコマンド識別コード(03H)

AL：デバイスタイプ(DA)(8*H)……*はチェックの対象にしない

出 力

CF：終了条件(0／1)

AH：00H

処 理

このコマンドによって，次の処理が行われます．

・ディスクコントローラの初期化．

・ユニットの接続状態のチェックと，READY状態の装置に対応するシステム共通領域(DISK EQUIP　55CH～55DH)のEquipment FlagをONにします．

・5インチハードディスク上のディップスウィッチの状態を読み取り，その情報をコントローラに通知します．

・リトラクト処理を行い，アームを不使用シリンダへ移動します．

ベリファイ

機　能

デバイスタイプ／ユニットナンバー(DA／UA)で指定されたデバイスの，指定されたディスクアドレスからデータを読み取り，読み取り動作ができることを確認します.

入　力

内部割り込みコード：0B1H

AH：BIOSコマンド識別コード(以下の8ビット)

7	6	5	4	3	2	1	0
*	*	r	*	0	0	0	1

他は「データ読み出し」と同じです.

出　力

「データ読み出し」と同じです.

処　理

ディスク上のデータが読み取れることを確認します.

センス

機　能

指定されたデバイスの状態を通知します.

入　力

内部割り込みコード：0B1H

AH：04H

AL：DA／UA(80H～81H)

出　力

CF：終了条件(0／1)

AH：以下のステータス

7	6	5	4	3	2	1	0	
d7	d6	d5	d4	0	0	0	0	…5MB
				0	0	0	1	…10MB
				0	0	1	1	…20MB

d4~d7については，ステータス一覧を参照してください．

代替トラックの指定

機 能

デバイスタイプ／ユニットナンバー(DA／UA)で指定されるデバイスの代替トラックアドレスを指定します．次項のFORMAT BAD TRACKコマンドによって，不良トラックの代替トラック処理が行われます．

入 力

内部割り込みコード：0B1H

AH：08H

AL ：DA／UA(80H~81H)

BX ：データ長(04H)

CX ：代替トラックのシリンダ番号

DH：代替トラックのヘッド番号

DL ：0

ES／BP：代替トラックアドレスを格納する4バイトバッファのアドレス

出 力

CF ：0

AH：00H

処 理

デバイスタイプ／ユニットナンバーにより，指定されるデバイスの，代替トラックアドレスをディスクコントローラに通知します．指定されたアドレスを指定された4バイトバッファに格納し，正常終了します．

不良トラックのフォーマット(FORMAT BAD TARCK)

機　能

デバイスタイプ／ユニットナンバー(DA／UA)で指定されたデバイスに対して，指定された不良トラックの代替トラックを割り付けます．代替トラックは，直前に代替トラック指定コマンドによって指定されたトラックです．不良トラックに代替トラックを割り付ける方法は次のように行います．

- 不良トラックの全セクタのID部に代替トラック有りのフラグを立て，データ部には直前の代替トラック指定コマンドで指定されたトラックアドレスを書き込みます．
- 代替トラックには，代替トラックフラグを全セクタのID部に立て，データ部にはE5Hを書き込みます．

入　力

内部割り込みコード：0B1H
AH：0BH
AL：DA／UA(80H～81H)
BX：データ長(4バイトバッファ-04H)
CX：不良トラックのシリンダ番号
DH：不良トラックのヘッド番号
DL：0
ES／BP：代替トラックアドレスの格納されている4バイトバッファの先頭アドレス

出　力

CF：終了条件(0／1)
AH：ステータス情報

処　理

不良トラックのフォーマットを行った後の代替トラック，不良トラックのREAD／WRITE／VERIFYコマンドの処理は次のようになります．

- 不良トラックへのREAD／WRITE／VERIFYコマンド
  自動的に代替トラックをアクセスします．
- 代替トラックへのREAD／WRITE／VERIFYコマンド
  エラーとなります．

2-3 グラフィックス

2-3-1 グラフィックBIOS

AH レジスタ	機能
0H	グラフィック画面の表示開始
41H	グラフィック画面の表示停止
42H	表示領域の設定
43H	パレットレジスタのセット
44H	ボーダーカラーのセット
45H	描画画面へのドットの書き込み
46H	描画画面からのドットの読み出し
47H	描画画面への直線，矩形の書き込み
48H	描画画面への円弧の書き込み
49H	描画画面へのグラフィック文字の書き込み
50H	描画タイミングモードの設定

表 2-3-1 グラフィック BIOS 機能一覧(コマンド識別コード：18 H)

注 意

・グラフィック BIOS 使用時は，スタックエリアを 30 バイト以上確保する必要があります．SS, SP をセットしなければなりません．

・CPU のステータスフラグのうち，IF と TF ビットを次のように設定しておきます．

IF：セット(割り込み可)

TF：クリア(シングルステップモードクリア)

・グラフィック BIOS 使用のために，描画などの情報の受け渡し，保存のために約 800 バイトの制御情報域が必要となります(UCW という)．これは，ユーザーがあらかじめ確保しなければなりません．

オフセット	ラベル	サイズ
0000	GBON_PTN	RB 1
0001	GBBCC	RB 1
0002	GBDOTU	RB 1
0003	GBDSP	RB 1
0004	GBCPC	RB 4
0008	GBSX1	RW 1
000A	GBSY1	RW 1
000C	GBLNG1	RW 1
000E	GBWDPA	RW 1

0010	GBRBUF	RW 3
0016	GBSX2	RW 1
0018	GBSY2	RW 1
001A	GBMDOT	RW
001C	GBCIR	RW 1
001E	GBLNG2	RW 1
0020	GBLPTN	RW 1
0020	GBDOTI	RB 8
0028	GBDTYP	RB 1
0029	GBFILL	RB 1
002A	GBGWK1	RW 1
002C	GBGWK2	RW 1
002E	GBGWK3	RW 1
0030	GBGWK4	RW 1
0032	GBGWK5	RW 1
0034	GBGWK6	RW 1
0036	GBGWK7	RW 1
0038	GBGWK8	RW 1
003A	GBGP122	RW 1
003C	GBGP34	RW 1
003E	GBGP56	RW 1
0040	GBGP78	RW 1
0042	GBGP910	RW 1
0044	GBGP1112	RW 1
0046	GBGP1314	RW 1
0048	GBGP1516	RW 1

注：GBLPTNとGBDOTIは同じオフセットを持ちます．

図 2-3-1　制御情報域

・矩形，円弧，グラフィック文字などの描画において，描画開始(終了)点の指定と描画方向の指定が必要です．直線，矩形の描画においては，開始点・終了点・描画方向の3つが矛盾しないように指定する必要があります．

描画開始方向ID	直　線	円　弧	グラフィック文字	矩　形
0				
1				
2				

注：ここで示している描画開始点，描画終了点は，それぞれ (GBSX1, GBSY1)，(GBSX2, GBSY2) の内容となる．
描画開始方向IDは，GBDSPの内容となる．

3				
4				
5				
6				
7				

●：描画開始点　○：描画終了点
▨：定義域　□：描画域
→：描画開始方向

表 2-3-2　描画方向

グラフィック画面の表示開始

機 能

グラフィック画面のCRTへの表示を開始します．テキスト画面とは独立して機能します．

入 力

内部割り込みコード：18 H

AH：40 H

出 力

全てのレジスタが保証されます．

処 理

GDCに対して，表示制御用のSTARTコマンド(0 DH)を送ります．FIFOバッファがFULLでなく，V-SYNCであることを確認した後，次のコマンドを送ります．

グラフィック画面の表示停止

機　能

グラフィック画面のCRTへの出力を停止します．テキスト画面とは独立して動作します．

入　力

内部割り込みコード：18 H

AH：41 H

出　力

全てのレジスタが保証されます．

処　理

GDCに対して表示制御用のSTOPコマンド(0 CH)を送ります．FIFOバッファがFULLでないことを確認した後，次のコマンドを送ります．

表示領域の設定

機　能

表示対象とする描画メモリ領域を設定します．使用するCRTのモード(モノクロ／カラー)と使用するグラフィックVRAMの領域を指定します．グラフィックVRAM領域の指定方法は，1つの表示画面に対応するVRAM領域がどのアドレスに対応するかを示すものです．VRAM上の32 Kバイト全体を1つの表示画面とする場合はALL，下位16 Kを1つの画面とする場合はLOWER，上位16 Kを1つとする場合はUPPERと呼び，この3つの指定区分のうちどれかを指定します．指定された条件に従って，GDCへ表示領域の設定を行います．

入　力

内部割り込みコード：18 H

AH：42 H

CH：CRTディスプレイモードとVRAMの指定(以下の8ビット)……640×400の時ALL，標準モードの時LOWERまたはUPPERを指定

7	6	5	4	3	2	1	0
V2	V1	M	B	0	0	0	0

V 2／V 1：VRAM 指定……01：UPPER
10：LOWER
11：ALL

M：モード指定……0：カラー
1：モノクロ

B：表示バンク指定……0：バンク 0 を使用
1：バンク 1 を使用(ただし，PC-9801／U では 0 に固定)

出　力

全てのレジスタが保証されます．

処　理

グラフィック関係の情報をモードレジスタにセットします．

・グラフィックモードとして，カラーかモノクロかをセットします(02 H，03 H)．
・走査線数として 200 本か 400 本かをセットします(09 H，08 H)．ただし，走査線数 200 の CRT に 400 本の指定をしても無意味です．
・1 行中の表示ライン数を，GDC に対して CSRFORM コマンドでセットします．

高解像度ディスプレイ　　1 ライン数／行
標準ディスプレイ　　　　2 ライン数／行

・表示開始アドレスおよび画面表示領域の大きさを GDC に対して SCROLL コマンドでセットします．

指　定　値	表示開始アドレス
LOWER	0
UPPER	16000(バイトアドレス)
ALL	0

装置タイプ	画面表示領域の大きさ
高解像度	400ライン
標準	200ライン

表 2-3-3　表示開始アドレス／画面表示領域の大きさ

＜グラフィック VRAM 領域と描画画面との関係＞

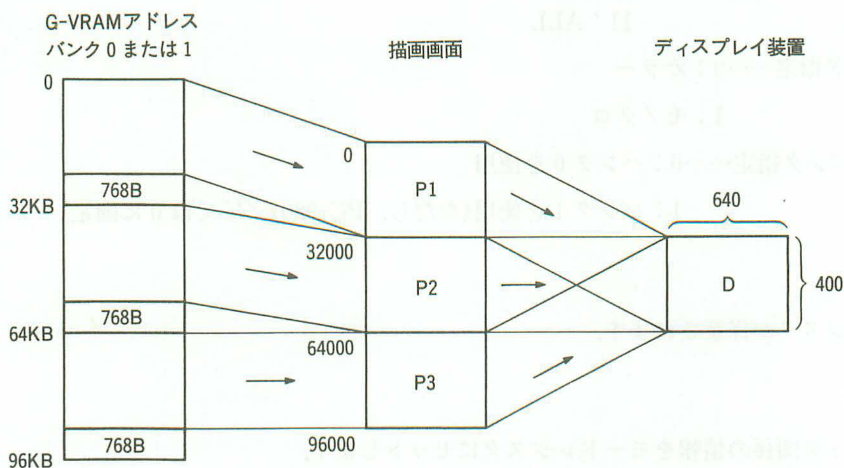


図 2-3-2　カラーモード（高解像度）

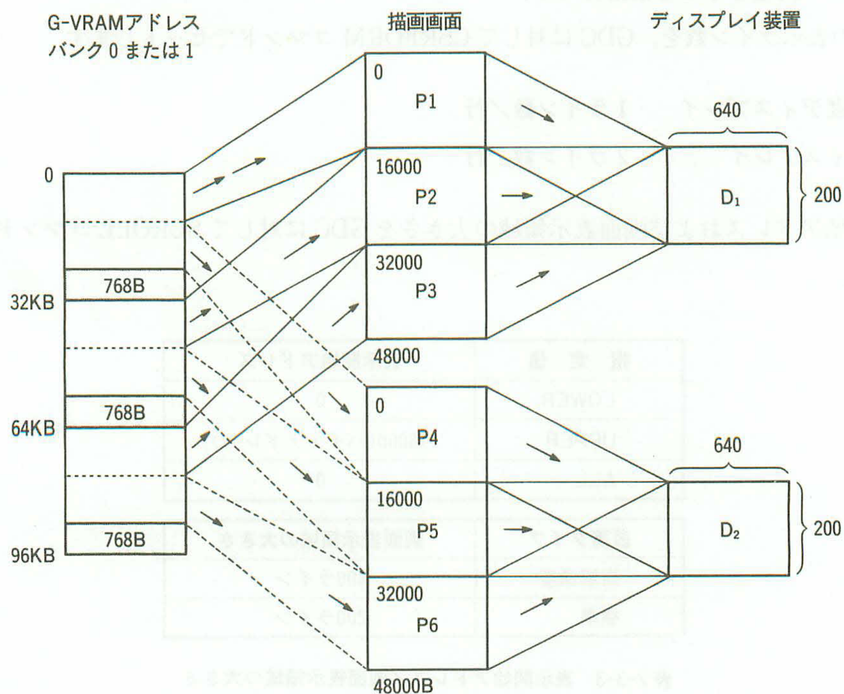


図 2-3-3　カラーモード（標準）

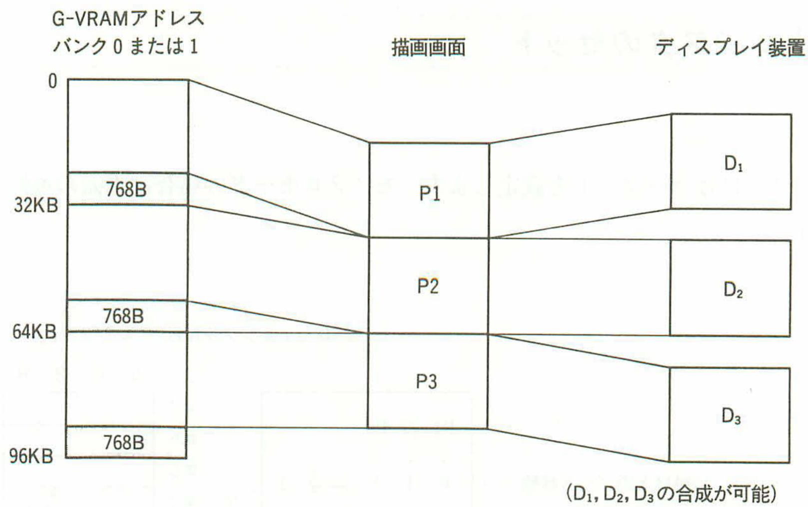


図 2-3-4 モノクロモード(高解像度)

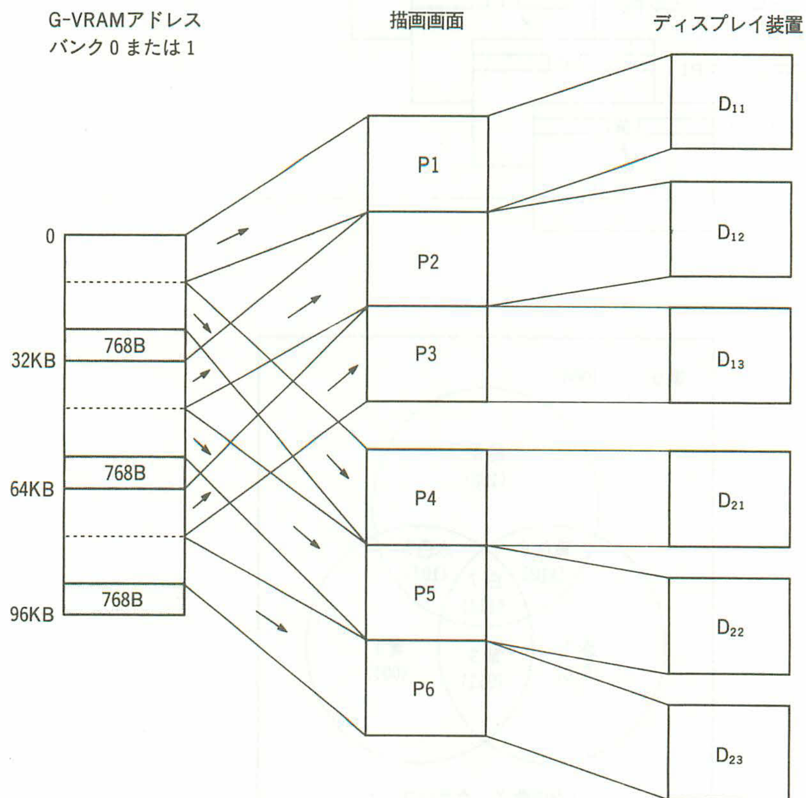


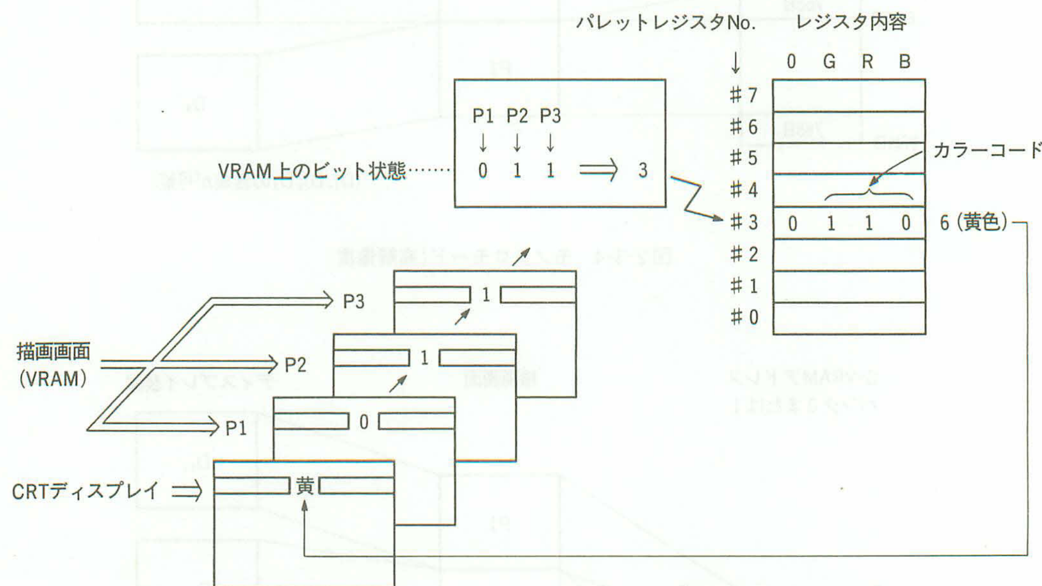
図 2-3-5 モノクロモード(標準)

注：モノクロモードにおける画面の選択・合成はパレットによって行います.

パレットレジスタのセット

機　能

パレットレジスタにカラーコードを設定します．モノクロモードの場合は画面の選択・合成の指定を行います．



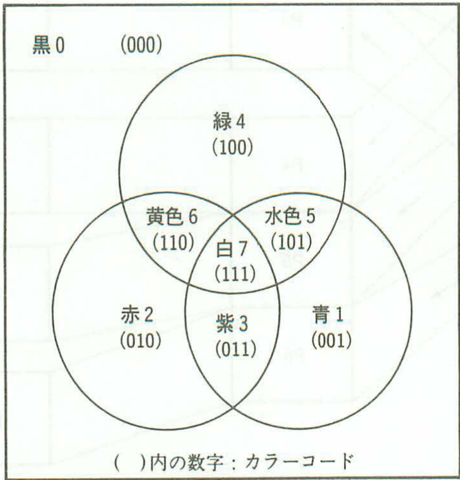


図2-3-6　パレットレジスタ

入　力

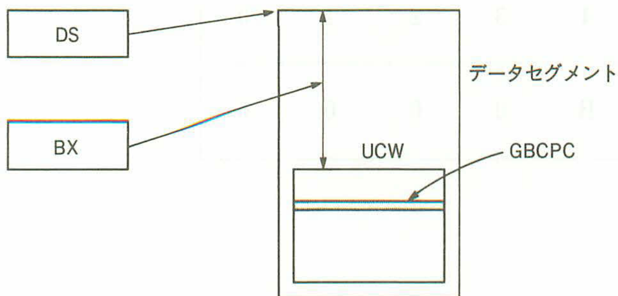
内部割り込みコード：18 H

AH：43 H

DS ：UCW のセグメントアドレス

BX：UCW のオフセットアドレス

UCW の GBCPC(オフセット 04 H，4 バイト)：カラーコード



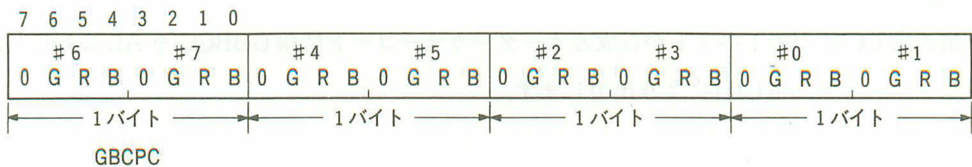


図 2-3-7　パレット入力

出　力

全てのレジスタが保証されます．

処　理

制御情報域 UCW 中の 4 バイトからなるカラーコード情報 GBCPC(8 エントリから成り，1 エントリ 4 ビット)を順次パレットレジスタに書き込みます．ライトパレットレジスタ(0 A 8 H，0 AAH，0 ACH，0 AEH)により，出力が行われます．1 回のパレットレジスタへの書き込みで，2 エントリ分のカラーコードを出力します．

ボーダーカラーのセット

機　能

標準ディスプレイを使用している場合には，バックグラウンドカラーと画面の境界のボーダーカラーを使用する事ができます．ボーダーカラーレジスタにカラーをセットすることにより，可能となります．

入　力

内部割り込みコード：18 H

AH：44 H

DS ：UCW のセグメントアドレス

BX ：UCW のオフセットアドレス

UCW の GBBCC(オフセット 01 H，1バイト)にセットするボーダーカラーコード

	7	6	5	4	3	2	1	0
GBBCC	0	G	R	B	0	0	0	0

出　力

全てのレジスタが保証されます．

処　理

制御情報域 UCW 中の1バイトから成るボーダーカラーコード情報 GBBCC を AL に転送し，ライトボーダーカラー(6 CH)により出力します．

描画画面へのドットの書き込み

機　能

指定された描画画面(G-VRAM)へドット単位の書き込みを行います．P 1，P 2，P 3 を個別に書き込む場合と P 1／P 2／P 3 の3画面を同時に書き込む場合の指定が可能です．「表示領域の設定」の項で述べた ALL，LOWER，UPPER の指定も必要です．また，P 4，P 5，P 6，P 4／P 5／

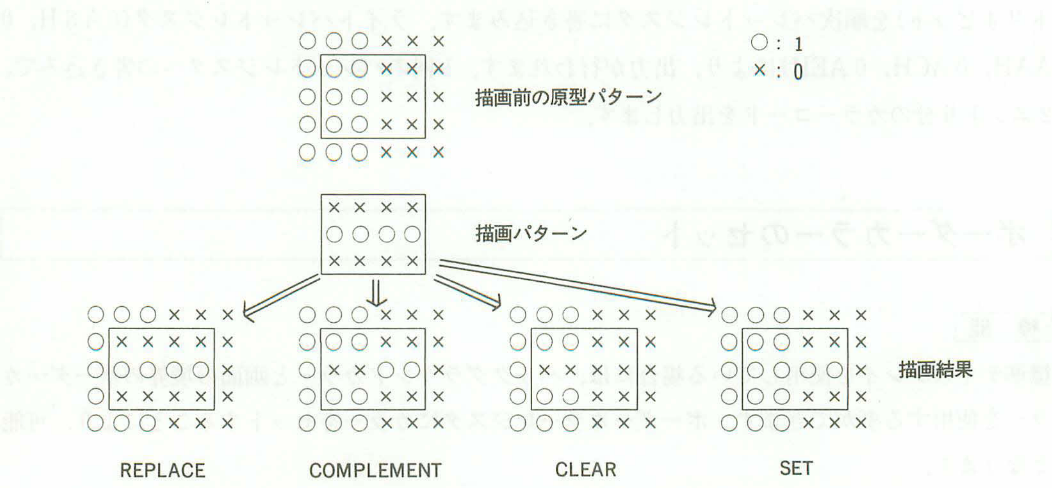


図 2-3-8　パターンのオペレーション

P 6の指定も可能です．単一描画画面への書き込みでは，それ以前の画面の状態と与えられた描画パターンとの間でオペレーション操作(Replace, Complemet, Clear, Set)を行い，その結果を書き込むことも可能です．

入 力

内部割り込みコード：18 H

AH：45 H

CH：対象とする描画画面の指定(以下の 8 ビット)

	7	6	5	4	3	2	1	0
CH	M	V	P 1	P 2	0	0	0	0

M：モード指定……0：標準
　　　　　　　　　1：高解像度

V：G-VRAM 指定……0：LOWER・ALL
　　　　　　　　　　1：UPPER

P 1／P 2：描画画面ナンバー……00～P 1(P 4)
　　　　　　　　　　　　　　　01～P 2(P 5)
　　　　　　　　　　　　　　　10～P 3(P 6)
　　　　　　　　　　　　　　　11～P 1／P 2／P 3(P 4／P 5／P 6)

b7	b6	b5	b4	描画画面と大きさ	
0	0	0	0	P1(0-18K)	16KB
		0	1	P2(32K-48K)	16KB
		1	0	P3(64K-80K)	16KB
		1	1	P1／P2／P3	48KB
0	1	0	0	P4(16K-32K)	16KB
		0	1	P5(48K-64K)	16KB
		1	0	P6(80K-96K)	16KB
		1	1	P4／P5／P6	48KB
1	0	0	0	P1(0-32K)	32KB
		0	1	P2(32K-64K)	32KB
		1	0	P3(64K-96K)	32KB
		1	1	P1／P2／P3	96KB

表 2-3-4　描画画面と大きさ

ES：描画パターンバッファのセグメントアドレス

DS：UCW のセグメントアドレス

BX：UCW のオフセットアドレス

UCW のコントロールワード

・GBON_PTN(1バイト)

3画面同時書き込み時の描画画面ナンバーと描画オペレーションモード指定

・GBDOTU(1バイト)

描画オペレーションモード指定(単一画面処理時のみ)

・GBX1(2バイト)

描画開始アドレスX座標(オリジナルスクリーン座標)

・GBY1(2バイト)

描画開始アドレスY座標(オリジナルスクリーン座標)

・GBLNG1(2バイト)

書き込みの長さ(ドット数)

・GBWDPA(2バイト)

描画パターンバッファの開始アドレス(オフセット)

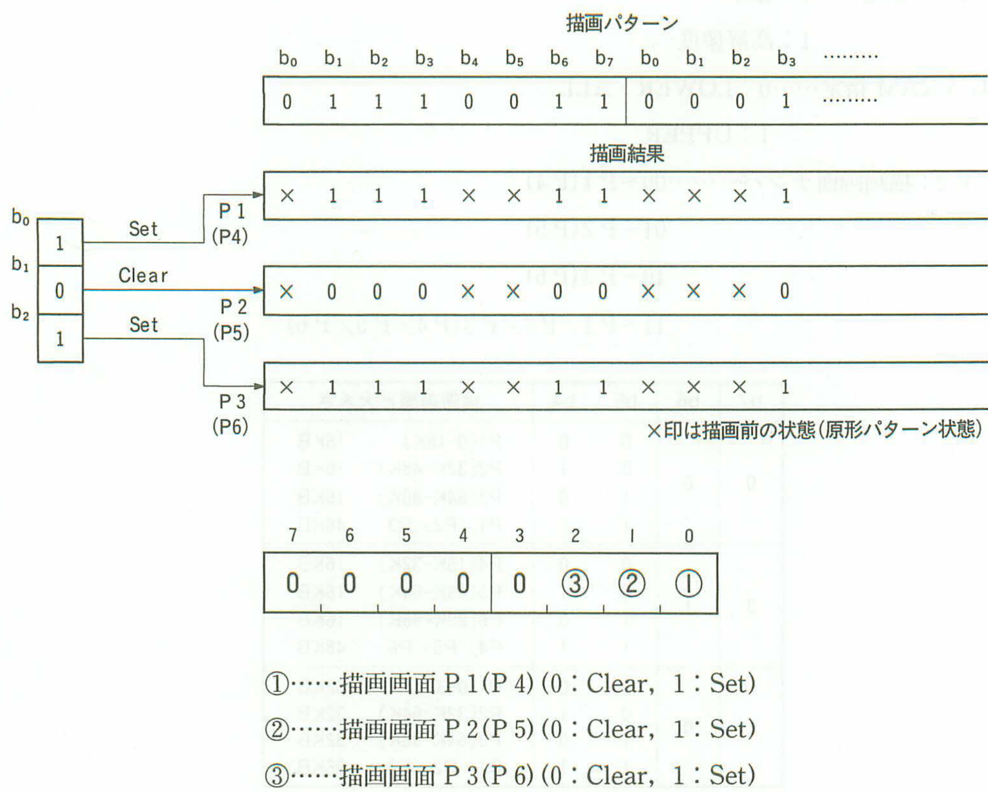


図2-3-9　GBON_PTN(オフセット0H)

注：3画面P1，P2，P3(またはP4，P5，P6)に対して同時書き込みを行う場合(CHのビット5，4が11)に描画オペレーションモードを指定します．

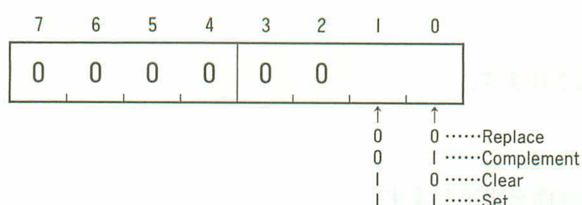


図 2-3-10 GBDOTU(オフセット 02 H)

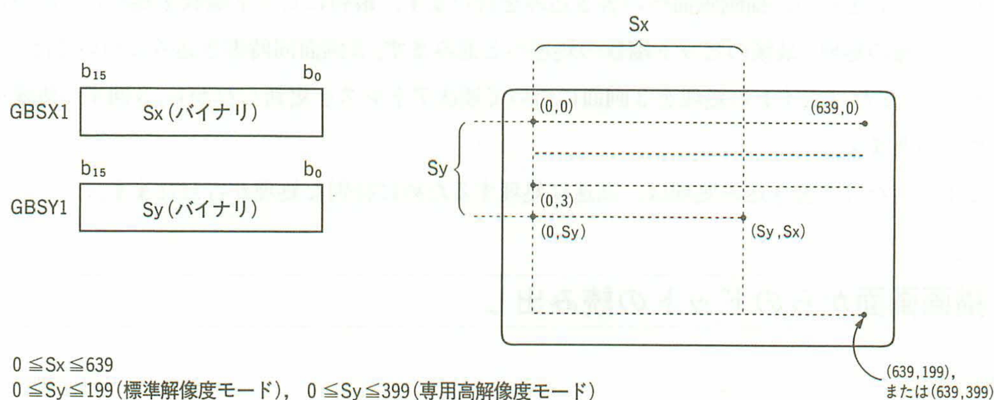


図 2-3-11 GBSX, GBSY(オフセット 08 H, 0 AH)

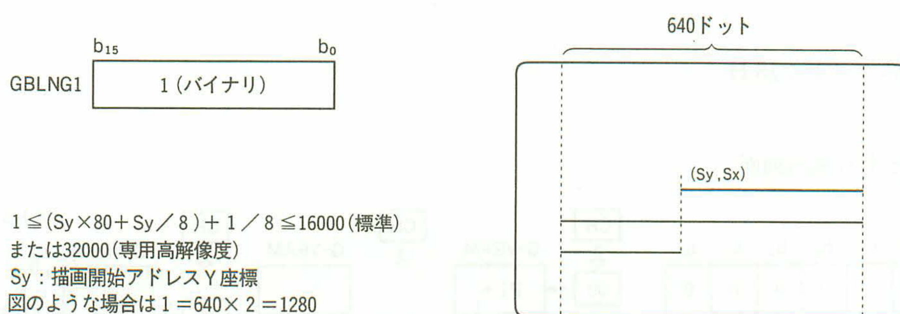


図 2-3-12 GBLNG 1(オフセット 0 CH)

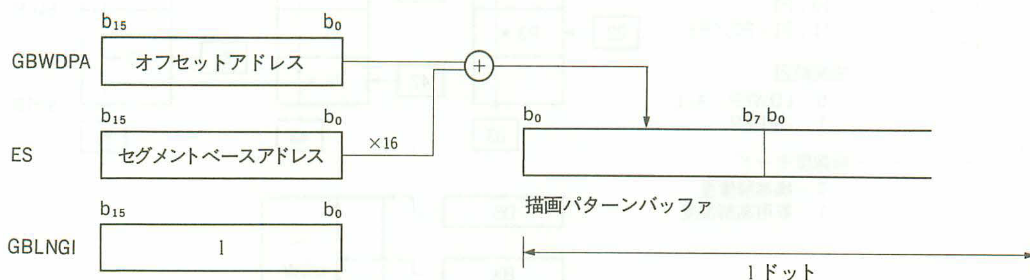


図 2-3-13 GBWDPA(オフセット 0 EH)

出　力

全てのレジスタが保証されます．

処　理

・CH の内容から描画画面を決定します．
・描画開始アドレスを計算し，最初のビット端数を決定します．
・3画面同時書き込みと単一画面書き込みとを区別して，描画パターンと描画画面との描画オペレーションを行い，描画画面への書き込みを行います．最初にビット端数を処理し，次からはバイト毎の処理，最後のビット端数の処理へと進みます．3画面同時書き込みについては，同一ビット，またはバイトの処理を3画面について順次アドレスを更新しながら，3回ずつ処理を進めていきます．
・1ドットだけの書き込み処理は，高速に処理するために特別な処理が行われます．

描画画面からのドットの読み出し

機　能

指定された描画画面(G-VRAM)から，指定されたバッファに対しドット単位の読み出しを行います．P1，P2，P3，P4，P5，P6の扱いは書き込みと同様です．

入　力

内部割り込みコード：18 H

AH：46 H

CH：対象とする描画画面

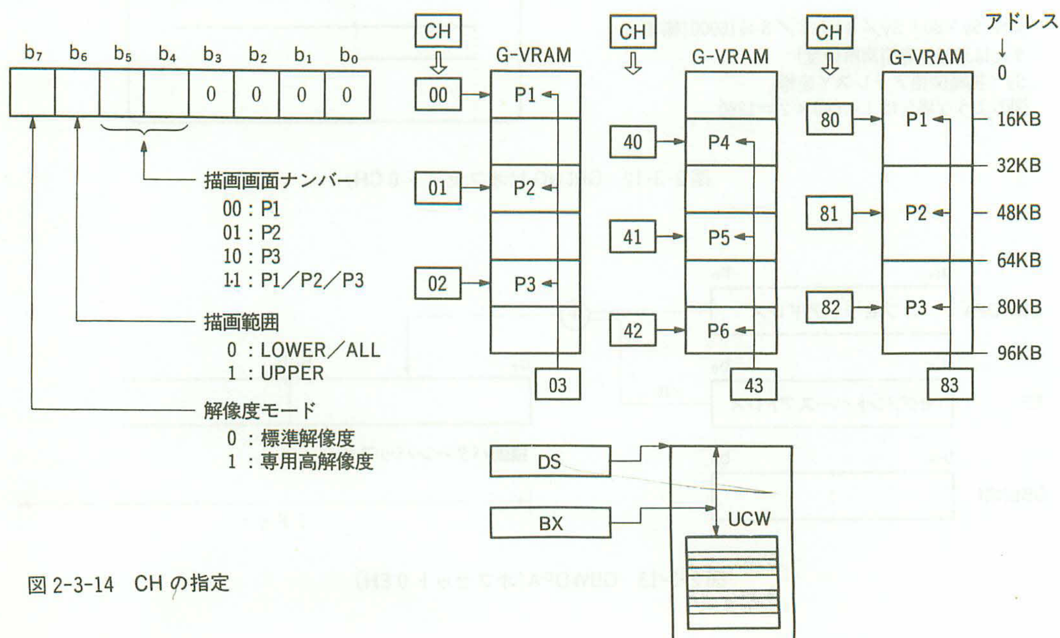


図 2-3-14　CH の指定

DS：UCWのセグメントアドレス
BX：UCWのオフセットアドレス
ES：読み込みバッファ(1～3)のセグメントアドレスを指定
UCWのコントロールワード

・GBSX 1, GBSY 1, GBLNG 1の扱いは「ドットの書き込み」と同じです.
・GBRBUF 1(2バイト)：読み込みバッファ1の開始アドレス
・GBRBUF 2(2バイト)：読み込みバッファ2の開始アドレス
・GBRBUF 3(2バイト)：読み込みバッファ3の開始アドレス
(単一画面読み込み処理はGBRBUF 1のみを使用します)

3画面同時読み込み処理の場合は，GBRBUF 1のポイント先からドット長までのバッファにP 1(P 4)画面からの読み込みデータが，GBRBUF 2のポイント先からドット長までのバッファにP 2(P 5)画面からのデータが，GBRBUF 3のポイント先からドット長までのバッファにP 3(P 6)画面からのデータが，それぞれ格納されます.

出　力

すべてのレジスタが保証されます.

処　理

「ドットの書き込み」と逆の処理を行います.

・CHレジスタより描画画面ナンバーを得ます. 場合によっては3画面同時読み出し処理であることを確認します.
・読み込み開始アドレスを計算し，最初の端数ビットのバッファへの読み出しを行います. バイト単位の転送を行い，最後の端数ビットの処理はバイト境界にそろえ，端数ビット以外のビット領域は不定となります.

描画画面への直線，矩形の書き込み

機　能

指定された描画画面(VRAM)上に直線(実線，破線等)または矩形を書き込みます. 書き込みの条件として，線種パターンを16ビットのパターンとして指定し，これと前の描画画面の状態との間でオペレーション操作(Replace, Complement, Clear, Set)を行った結果を書き込みます. また，単一画面への書き込み，3画面同時書き込みのどれでも使用可能です.

描画パターン		0 0 1 1
描画画面の前の状態		0 1 0 1
描画画面の結果	Replace	0 0 1 1
	Complement	0 1 1 0
	Clear	0 1 0 0
	Set	0 1 1 1

表 2-3-5　オペレーション操作

入　力

内部割り込みコード：18 H

AH：47 H

CH：対象とする描画画面の指定(指定方法は「ドットの読み出し」と同じ)

DS：UCW のセグメントアドレス

BX：UCW のオフセットアドレス

UCW のコントロールワード

・GBON_PTN(8 ビット)
　3 画面同時書き込み時の描画画面ナンバーと描画オペレーションモードの指定
・GBDOTU(8 ビット)：単一画面処理時の描画オペレーションモード
・GBDSP(8 ビット)：描画開始方向
・GBSX 1，GBSY 1(各 16 ビット)：描画開始アドレス　X，Y 座標
・GBSX 2，GBSY 2(各 16 ビット)：描画終了アドレス　X，Y 座標
・GBLPTN(16 ビット)：線種パターン
・GBDTYP(8 ビット)：描画タイプ

① GBON_PTN(オフセット 0 H)

3 画面同時書き込みの場合に使用する描画オペレーションモードの指定を行います．

7	6	5	4	3	2	1	0
0	0	0	0	0	P 3	P 2	P 1

P 3：画面 P 3(P 6)，P 2：画面 P 2(P 5)，P 1：画面 P 1(P 3)のそれぞれには，Clear なら 0 を Set なら 1 を指定します．3 画面 P 1／P 2／P 3(または P 4／P 5／P 6)に対して同時書き込みを行う場合(CH の 5 ビットと 4 ビットが 11 の場合)に描画オペレーションモードは次のような働きをします．線種パターンは 16 ドット単位で繰り返されます．

図 2-3-15 GBON_PTN

② GBDOTU(オフセット 02 H)

単一画面書き込みの場合に使用する描画オペレーションモードの指定を行います.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	O2	O1

O2, O1ビットがそれぞれ次のように対応します.

00 : Replace

01 : Complement

10 : Clear

11 : Set

オペレーションモード	描 画 結 果
Replace	描画パターンをそのまま置き換える
Complement	描画前のパターンと描画パターンのXORをとる
Clear	描画パターン: 1……0にクリア 0……変化なし
Set	描画パターン: 1……1にセット 0……変化なし

Replaceでは描画前のVRAM内容には関係なく, 新しく設定した描画パターンが描画されます. ほかの3つのモードでは描画パターンが "1" であるドットに対してのみ, 1/0反転, 0設定の修正を加え, "0"であるドット位置に対応するVRAMの内容は再書き込みされます.

③ GBDSP(オフセット 03 H)

描画方向の指定(00 Hから07 Hまでの数値を指定)を行います.

7	6	5	4	3	2	1	0
0	0	0	0	0	S3	S2	S1

④ GBSX 1，GBSY 1(オフセット 08 H，0 AH)

描画開始アドレス(オリジナルスクリーン座標)の指定を行います．X, Y はオリジナルスクリーン座標で描画開始点のドットの位置を示します．

15・・・・・・・・・・・・・・0

GBSX 1 X　　(バイナリ)　　0≦X≦639

GBSY 1 Y　　(バイナリ)　　0≦Y≦199(標準解像度)，399(高解像度)

⑤ GBSX 2，GBSY 2(オフセット 16 H，18 H)

描画終了アドレスの指定を行います．扱いはGBSX 1，GBSY 1と同じです．

⑥ GBLPTN(オフセット 20 H)

線種パターンの指定を行います．

0・・・・・・・・・・・・・・15

GBLPTN　・・・・・・・・・・・・・・・・・

描画オペレーションの描画パターンになるものです．16 ビットパターンで繰り返し使用されます．GBLPTN の内容が描画パターンそのものです．

⑦ GBDTYP(オフセット 28 H)

描画タイプの指定を行います．

	7	6	5	3	2	1	0
GBDTYP	0	0	0	0	T 3	T 2	T 1

T 3，T 2，T 1は次のように指定します．

001：直線

010：矩形

100：円弧

出　力

全てのレジスタが保証されます．

処　理

この処理は後述の「円弧の書き込み」のルーチンと共通しています．直線，矩形，円弧について GDC に対するコマンド指示の概要を下に示します．

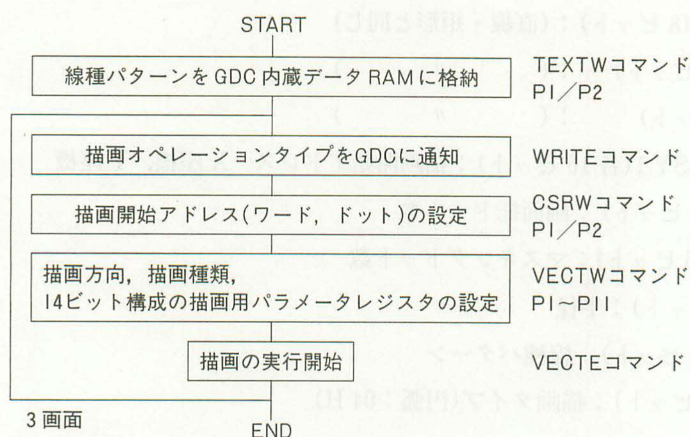
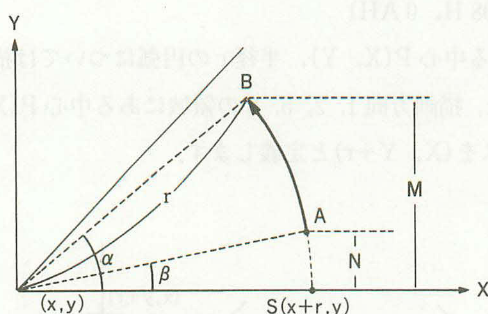


図 2-3-16　GDC に対するコマンド指示

描画画面への円弧の書き込み

機　能

指定された描画画面(VRAM上)に円弧(円は8個の円弧の集まりとして扱います)を書き込みます．書き込みの条件として，直線，矩形の描画と同様に描画オペレーション操作の指定，半径，マスキングドット数，描画総ドット数，描画開始方向を与える必要があります．



半径：r
描画開始点S：(x+r,y)
マスキングドット数：N
描画開始方向：A→B
描画総ドット数：M
線種パターン
描画オペレーションモード

図 2-3-17　円弧描画条件

入　力

内部割り込みコード：18 H

AH：48 H

CH：対象とする描画画面の指定(「直線，矩形の書き込み」参照)

DS：UCW のセグメントアドレス

BX：UCW のオフセットアドレス

UCW のコントロールワード

・GBON_PTN(8ビット)：(直線・矩形と同じ)
・GBDOTU(8ビット)　：(　　〃　　)
・GBDSP(8ビット)　　：(　　〃　　)
・GBSX 1／GBSY 1(各16ビット)：描画開始アドレス　X座標，Y座標
・GBLNG 1(16ビット)：画面総ドット数
・GBMDOT(16ビット)：マスキングドット数
・GBCIR(16ビット)：半径
・GBLPTN(16ビット)：線種パターン
・GBDTYP(8ビット)：描画タイプ(円弧：04 H)

①，②は「直線，矩形の書き込み」の項を参照して下さい．

③ GBDSP

直線・矩形と共通ですが，円弧描画の場合は1／8弧描画単位ですから，描画方向とはどの領域に含まれる1／8弧であるかを指定する事になります．次の0～7までの領域をコード0～7で指定します．

0：180°～225°　1：45°～90°　2：270°～315°
3：135°～180°　4：0°～45°　5：225°～270°
6：90°～135°　7：315°～360°(0°)

④ GBSX 1，GBSY 1(オフセット08 H，0 AH)

描画方向0，3，4，7の領域にある中心P(X，Y)，半径rの円弧については描画開始アドレスは(X+r，Y)と定義します．また，描画方向1，2，5，6の領域にある中心P(X，Y)，半径rの円弧については描画開始アドレスを(X，Y+r)と定義します．

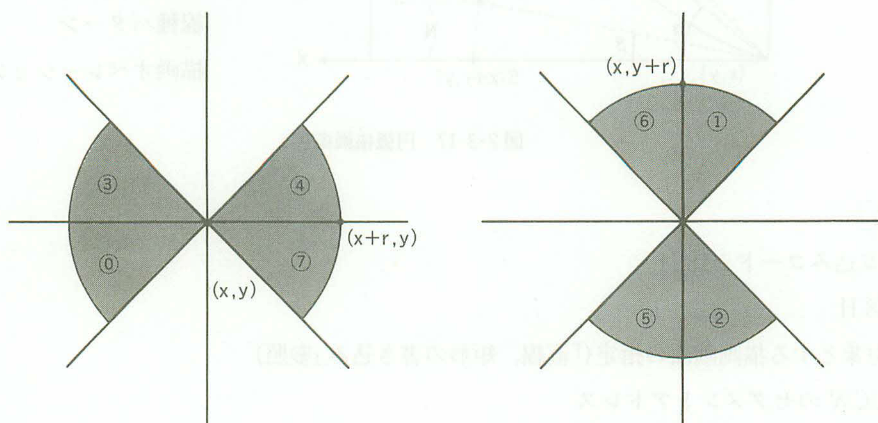


図2-3-18　X，Y座標

GBSX 1，GBSY 1の内容はそれぞれオリジナルスクリーン座標で表され，次の範囲となります．

0≦(GBSX 1)≦639

0≦(GBSY 1)≦199(標準解像度)，399(高解像度)

⑤ GBLNG 1(オフセット 0 CH，描画総ドット数)及び，GBMDOT(オフセット 1 AH，マスキングドット数)

	15　　　　　　0	
GBLNG1	バイナリ：m	m：描画総ドット数をバイナリで表します．
GBMDOT	バイナリ：n	n：マスキングドット数をバイナリで表します．

図 2-3-19　GBLNG 1／GBMDOT

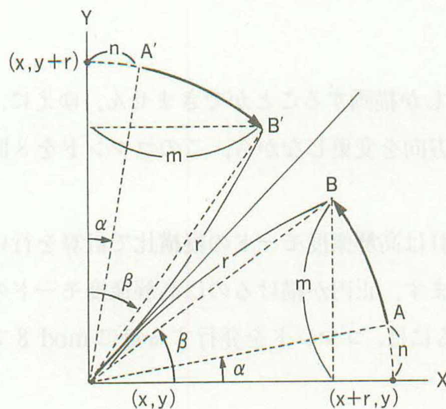


図 2-3-20　m，n 定義図と補足

上図において，中心(x, y)，描画開始アドレス(x+r, y)または(x, y+r)描画開始軸からみた開始角 α，終了角 β としたとき，孤 A → B，または A′→ B′の描画ドット数 m，マスキングドット数 n を次のように定義します

m=r sinβ

n=rsinα

実際にはこのような値の整数値を指定します．ここで，描画開始軸とは孤が存在する領域が⓪③④⑦の場合は x 軸，①②⑤⑥の場合は y 軸となります．

描画総ドット数 m と半径 r の間には次のような関係があります．

0≦m≦［r／√2］　(［］は小数点を切り上げた最小の整数)

⑥ GBCIR(半径)



図2-3-21　GBCIR

⑦ GBLPTN(「直線，矩形の書き込み」参照)

⑧ GBDTYP(描画タイプ)

```
15            0
GBDTYPE   0・・・・・・・・・・・0100   04 H(円弧を指定します)
```

出　力

全てのレジスタが保証されます

処　理

直線・矩形と同様の処理を行います．

注　意

- GDCでは1回のコマンドで1／8円弧しか描画することができません．ゆえに，円を描画するときには描画開始アドレス，描画開始方向を変更しながら，このコマンドを8回実行することによって行います．
- このコマンドで描画する円弧(または円)は高解像度モードの縦横比で計算を行います．ゆえに，標準解像度モードでは楕円が表示されます．正円が描けるのは高解像度モードのみです．
- 円を描画するために描画方向を指定するには，コマンドを発行する毎にmod 8で5ずつ増分する事により可能です．

描画画面へのグラフィック文字の書き込み

機　能

指定された描画画面(VRAM上)にグラフィック文字を書き込みます．描画するグラフィック文字は与えられた基本パターン情報(8×8ドットまたは8×8ドットより小さいもの)によって与えられた領域に描画オペレーションを行いながら書き込みます．描画すべき領域が基本パターン情報よりも大きいと，基本パターン情報により繰り返し全ての描画すべき領域に対して操作していきます．このコマンドは8×8ドットより大きな領域を同一の8×8ドットのパターンで塗りつぶす場合などに有効です．

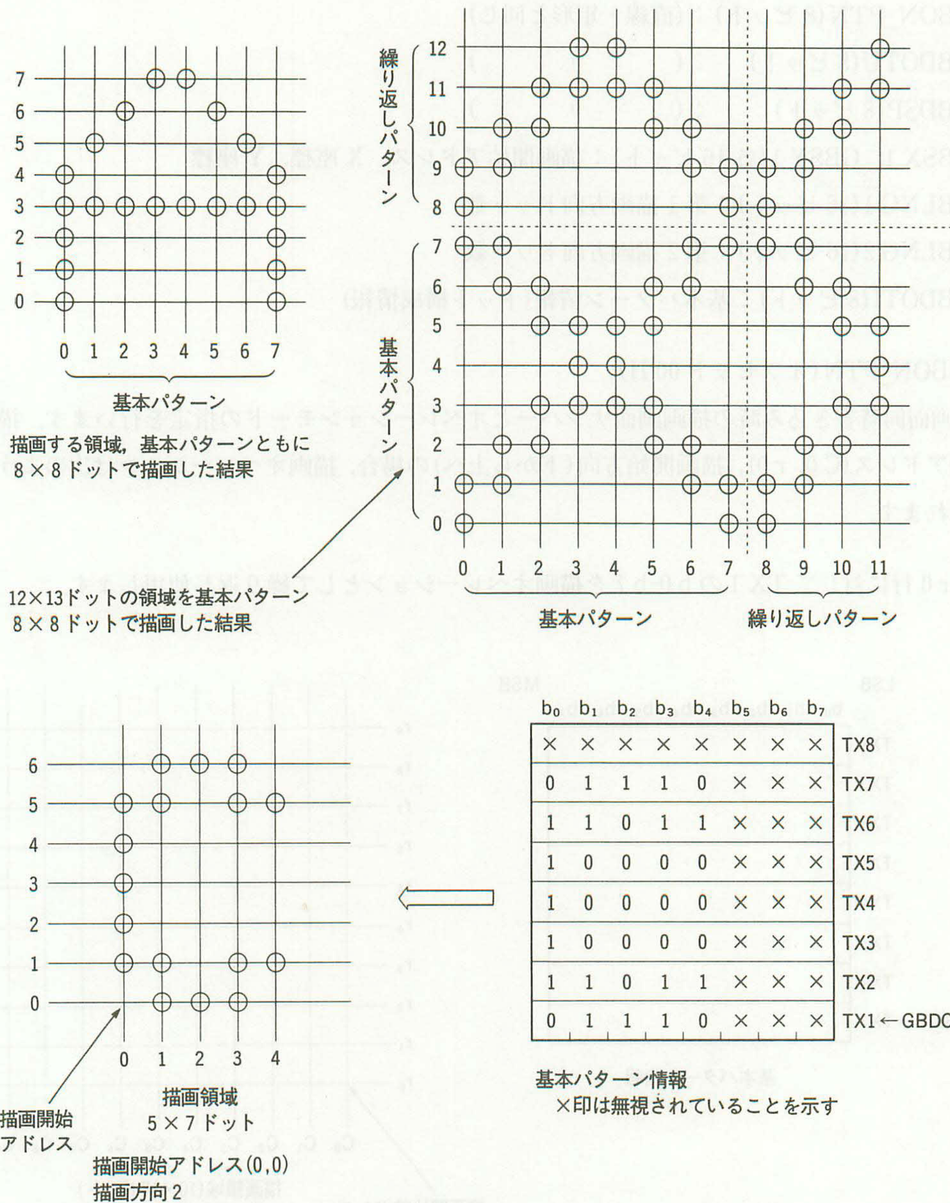


図 2-3-22 グラフィック文字の描画

入 力

内部割り込みコード：18 H

AH：49 H

CH：対象とする描画画面の指定(「直線，矩形の書き込み」参照)

DS ：UCW のセグメントアドレス

BX：UCW のオフセットアドレス

UCW のコントロールワード

・GBON_PTN(8ビット)：(直線・矩形と同じ)
・GBDOTU(8ビット)　：(　〃　)
・GBDSP(8ビット)　：(　〃　)
・GBSX 1，GBSY 1(各16ビット)：描画開始アドレス　X座標，Y座標
・GBLNG 1(16ビット)：第1描画方向ドット数
・GBLNG 2(16ビット)：第2描画方向ドット数
・GBDOTI(8ビット)：基本パターン情報(ドット構成情報)

① GBON_PTN(オフセット00 H)

3画面同時書き込み時の描画画面ナンバーとオペレーションモードの指定を行います．描画開始アドレス(C 0，r 0)，描画開始方向(下から上へ)の場合，描画オペレーションは次のように行われます．

・r 0行に対してTX 1のb 0-b 7を描画オペレーションとして繰り返し使用します．

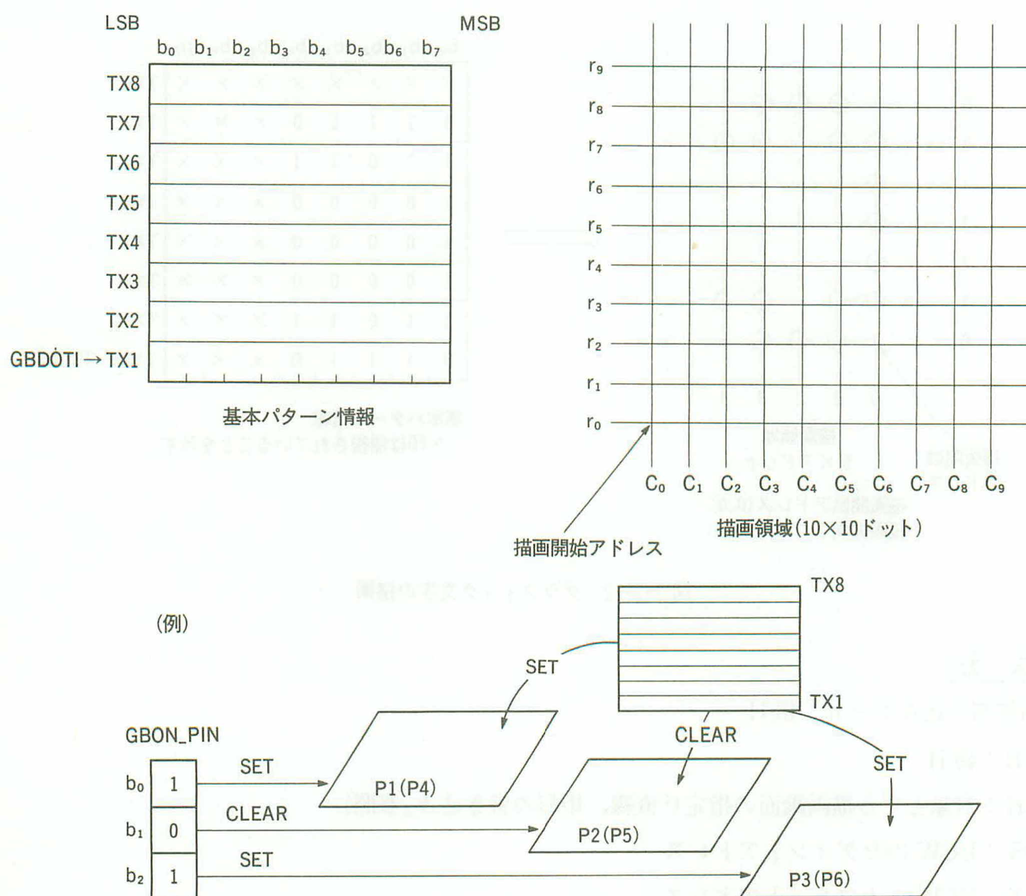


図2-3-23　GBON_PTN

・次にr1行に対してTX2のb0-b7を描画オペレーションとして繰り返し使用します．この様に順次，r1，r1+1と進めます．
・r8行，r9行……になるともう一度TX1，TX2，……と繰り返し使用します．

② GBDOTU(オフセット02H)

③ GBDSP(オフセット03H)

④ GBSX1．GBSSY1(オフセット08H，0AH)

扱いは「直線，矩形の書き込み」と同じです．

⑤ GBLNG1(オフセット04H)

第1描画方向ドット数(x)の指定を行います．

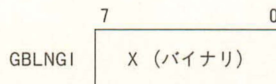


図2-3-24 GBLNG1

・8×8ドットの領域に対する場合は0
・8×8ドット以外の領域に描画する場合は横方向のドット数

⑥ GBLNG2(オフセット1EH)

第2描画方向ドット数(y)の指定を行います．

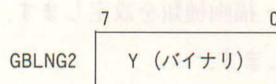


図2-3-25 GBLNG2

・8×8ドットの領域に対する場合は0
・8×8ドット以外の領域の場合は(縦方向のドット数-1)

⑦ GBDOTI(オフセット20H)

基本パターン情報の指定を行います．グラフィック文字描画時に描画領域に対して描画オペレーションを行うためのドットパターンを持つ8バイト情報です．この8バイト情報はGDCのコマンドであるTEXTWコマンドによって，GDC内臓データRAMに格納されます．GBDOTIからGBDOTI+7の順にGDC内臓データRAMのアドレスTX8からTX1へTEXTWコマンドの8個のパラメータによって格納されます．

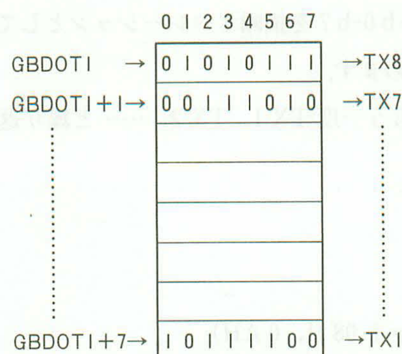


図2-3-26　GBDOT1

描画時にはTX1からTX8の順に参照されます．

出　力

全てのレジスタが保証されます．

処　理

GDCに対するコマンド指示は次のとおりです．

- TEXTWコマンドによって，基本パターン情報をGDCに送ります．
- WRITEコマンドによって，描画オペレーションモードを設定します．
- CSRWコマンドによって，描画領域を設定します．
- VECTWコマンドによって，描画方向・描画種類を設定します．TEXTEコマンドによって，グラフィックス文字描画の実行を開始します．

描画タイミングモードの設定（高速書き込みモードの指定）

機　能

描画画面に対するGDCからの書き込み（描画）タイミングには2つのタイミングがあります．

①フラッシュレス画面
　CRTへの表示時間とVRAMメモリリフレッシュ期間を除いた期間を書き込みタイミングとします．
②フラッシュ画面（高速書き込みモード）
　CRTへの表示中でもVRAMへの書き込みを可能にします．画面上にフラッシュが発生しますが，1に比べて書き込み速度は約5倍になります．

このコマンドは①か②のどちらかを選択するためのものです．

入　力

内部割り込みコード：18 H

AH：4 AH

CH：描画タイミングモードの設定……06 H：フラッシュ

16 H：フラッシュレス

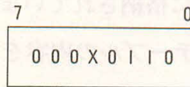


図 2-3-27　CH の設定

X：描画タイミングモード……0：フラッシュ

1：フラッシュレス

出　力

全てのレジスタが保証されます．

処　理

・GDC の動作制御のための SYNC コマンドを発行して，GDC に指示を行います．
・CRT が表示中かどうかを調べ，つぎのコマンドを実行します．

```
CRT ON
    MOV     AL, 0FH
    OUT     0A2H, AL
    MOV     AL, CH(06H or 16H)
    OUT     0A0H, AL
```

```
CRT OFF
    MOV     AL, 0EH
    OUT     0A2H, AL
    MOV     AL, CH(06H or 16H)
    OUT     0A2H, AL
```

図 2-3-28　CRT コマンド

2-3-2　グラフ LIO

N 88-日本語 BASIC(86) version 3.0 でサポートしている PC 9801 U／UV／VF／VM 用拡張グラフィック機能は BASIC 側でサポートしているものであり，この項では解説しません．

2-3-2-1　グラフ LIO の位置づけ

①用途

・N 88-BASIC グラフ機能を実現します．
・CP／M-86，MS-DOS 上の機械語プログラムからの使用が可能です．
・N 88-BASIC システム上の機械語プログラムからの使用が可能です．

②制御上の位置づけ

グラフィック BIOS およびテキスト BIOS は CRT 関係ハードウェアを直接に制御し，より上位のプログラムからの使用を容易なものにするためにあります．グラフ LIO はこれらの BIOS を使用して，よりロジカルなグラフィック処理機能を実現します．

③実際上の位置づけ

・グラフ LIO は ROM に実装されています．

・グラフ LIO は 17 種類のコマンド(後述)からなり，グラフ LIO のコマンドを処理するコマンドプロシージャは割り込みベクタを介して呼び出されます．

・グラフ LIO のコマンドプロシージャに対するエントリポイントは1つのシステムテーブルとして，グラフ LIO モジュールの特定の領域に格納されています．上位のプログラムがグラフ LIO を使用する場合は，まずこのエントリテーブルの内容を，使用する割り込みベクタ設定する必要があります．LIO コマンドプロシージャエントリポイントテーブルはメモリアドレス F 9900 H から始まる ROM 上に書き込まれており，次のような形式をしています．

F9900H（エントリ数）

+0	11			
+4	A0	00		GINITコマンド
+8	A1	00		GSCREENコマンド
+12	A2	00		GVIEWコマンド
+16	A3	00		GCOLOR1コマンド
+20	A4	00		GCOLOR2コマンド
+24	A5	00		GCLSコマンド
+28	A6	00		GPSETコマンド
+32	A7	00		GLINEコマンド
+36	A8	00		GCIRCLEコマンド
+40	A9	00		GPAINT1コマンド
+44	AA	00		GPAINT2コマンド
+48	AB	00		GGETコマンド
+52	AC	00		GPUT1コマンド
+56	AD	00		GPUT2コマンド
+60	AE	00		GROLLコマンド
+64	AF	00		GPOINT2コマンド
+68	CE	00		GCOPYコマンド
+70	00	00		GRAPH BIO
+72	ID情報		エントリポイントのオフセットアドレス	

図 2-3-29　エントリ表

＊エントリポイントのセグメントベースは F 990 H です．

＊セグメントレジスタへの格納値は F 990 H です．

＊ N 88-BASIC システムでの割り込みベクタ番号は A 0〜AF，CE を使用します．

2-3-2-2　グラフ LIO の機能概要

①図画描画機能

論理座標(-23768～32767 までの整数を使用した X, Y 座標)で表現した直線・楕円・矩形・領域の塗りつぶし・ドット等を描く機能です.

②表示制御機能

CRT に実際に表示するための G-VRAM 上の画面制御. つまり, 画面合成や画面モードに従った表示画面の制御, 実際に CRT に表示するための制御を行います.

③描画領域制御機能

G-VRAM 上の図形等を合成する領域(ビューポート)に対する制御を行います.

④画面情報の退避・復旧の制御

画面情報をメモリ上の別の領域に格納したり, そこから戻したりする機能です.

2-3-2-3　グラフ LIO 使用上の概念

①画面モード

画面モードは次に示すように 4 つのモードがあり, これらは場合により色表示の区別だけで制御を分けたり, 解像度の区別だけで制御を分ける時もあります.

	モノクロモード	カラーモード
標　準	グラフィックモード	カラーグラフィックモード
高解像度	高解像度ディスプレイモード	高解像度カラーディスプレイモード

表 2-3-6　画面モード

②グラフ画面

1 で述べたモードにより, 使用可能な画面数が異なります. モードによる違いを下の表に示します.

	グラフィック	カラーグラフィック	高解像度ディスプレイ	高解像度カラーディスプレイ
画面数	(2組の3画面) ×2対 ＝12画面	(2組の1画面) ×2対 ＝4画面	(1組の3画面) ×2対 ＝6画面	(1組の1画面) ×2対 ＝2画面
合　成	組の中の合成	不　可	組の中の合成	不　可

表 2-3-7　グラフ画面

* PC 9801 U の場合は 1 対です. 16 色グラフィックボード使用の場合は, グラフィックモードは 16 画面, 高解像度ディスプレイ画面は 8 画面まで使用できます.

③アクティブ画面とディスプレイ画面

・アクティブ画面とは描画対象画面の事で，画面番号で指定します．

・ディスプレイ画面とはディスプレイに表示する画面の事で，単一画面または合成画面の識別コードによって指定します．

④ LIO 論理座標系

論理座標系とは-32768 から 32767 までの整数値をとる X，Y で指定される座標系で，論理的な図形描画を行います．実際に描画が行われ，表示が可能なのは次の図のアクティブ画面内です．

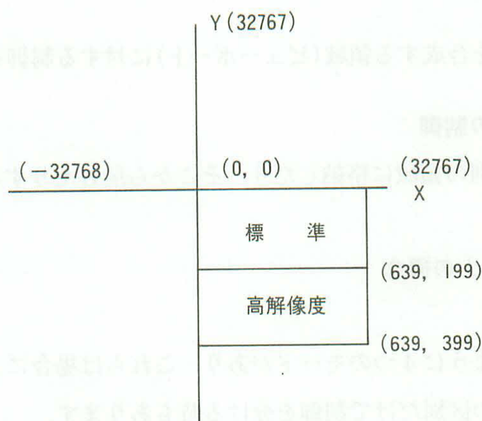


図 2-3-30　LIO 論理座標

⑤カラー指定

図形描画におけるカラー指定はパレット番号と呼ばれる論理的な色で示します．これは0から7までの整数で，パレット番号は3画面(16色拡張ボード使用時は4画面)のビット状態で表されます．モノクロディスプレイの場合，パレット番号が0ならば黒，それ以外なら白と扱われ，1画面のビット状態で黒・白が表現されます．

⑥ビューポート

論理座標系のアクティブ画面内の描画領域の事をいいます．実際の描画機能はビューポート内にのみ反映され，ビューポートを使用していない場合はアクティブ画面全体を描画領域とします．

2-3-2-4　グラフ LIO 使用のための準備

①ワークエリアの確保

グラフ LIO を使用するためには，各コマンドプロシージャで使用するワークエリアを確保する必要があります．このエリアを GCOPY コマンド使用時には 1400 H バイト，それ以外は 1200 H バイト確保する必要があります．このエリアは直接に使用することはできません．指定方法

は，LIOコマンド呼び出し時にDSレジスタで指定します．データセグメントに位置付けられ，データセグメント上の相対0番地からとられます．DSの内容は次のようになっています．数字は相対アドレスの始めを表しています．未使用領域はプログラムで使用することは可能ですが，他の領域への干渉を防ぐ事が必要です．

未使用領域	グラフ LIO 共同作業域 128B	未使用	グラフ LIO 共同作業域 56B	未使用	グラフ LIO 個別作業域 512B	未使用	GCOPY 作業域 128B
＋0H	＋620H		＋A08H		＋1000H		＋1380H

表 2-3-8 ワークエリア

②グラフLIOスタックエリアの確保

グラフLIO使用時にはスタックエリアとして，約128バイトのスタックエリアを確保する必要があります．このスタックエリアは，グラフLIO内でのスタック情報と，グラフLIOからグラフBIOまたはテキストBIOを使用するために，それぞれのBIOで使用するスタックエリアとを合わせたエリアが必要です．グラフLIO固有に64 B，グラフBIOで30 B，テキストBIOで約32 Bです．

③割り込みベクタの設定

グラフLIOのモジュールの先頭にエントリポイントテーブルがあるので，この値を設定しようとする割り込みベクタのオフセットアドレスに格納します．また，セグメントベースはF 990 Hです．なお，これはN 88-BASICの場合であり，CP／MやMS-DOS上で使用する場合は，割り込みベクタが使用されている状況を確認した上で，未使用の割り込みベクタにこれらの値を設定する事ができます．N 88-BASICで使用する場合は，エントリポイントテーブルのID情報が，割り込みベクタ番号に対応するように設定されています．

④内部割り込みコード0C5Hに対応する割り込み処理ルーチンの作成と登録

・グラフLIOでは，比較的時間のかかる描画処理を行っている場合に，処理の途中で中断を可能にするために一定の処理ごとに，0C5Hの内部割り込みルーチンのコールを行っています．ここでは，中断したときの画面情報の退避や，STOPキーが押された時の画面情報の扱いについてグラフLIO使用者が必要に応じた対応ができるようになっています．

・割り込みベクタ0C5H番号0C5Hの割り込みベクタの内容は必ず対応する割り込み処理ルーチンのエントリポイントを指すようにして下さい．

・注意事項として，グラフLIOルーチンで使用しているレジスタを保証すること，本ルーチンからグラフLIOルーチンの再呼び出しの禁止，作業域の保証，グラフLIOで使用するハード／ソフトのリソースの変更の禁止，などがあります．

・割り込み処理ルーチンは例えば，IRETのみのルーチンであっても構いません．

・0C5Hに対応する割り込み処理ルーチンをSTOPキーのチェックに使用することができます．



図2-3-31 STOPキーのチェック

2-3-2-5 グラフLIOの使用法概略

①初期設定

● GINTコマンドの呼び出し

各種リソースの初期設定を行うため，GINTコマンドを送ります．DSはグラフLIOワークエリアの先頭アドレス0として設定します．SS／SPはグラフLIOスタックエリアを設定するために使用します．

●グラフLIOコマンドの呼び出し方法

DSはGINTで指定したDSと同じ値を使用し，パラメータリストはDSによって示されたデータセグメント内に作成します．パラメータリストの先頭オフセットアドレスはBXで示します．スタックアドレスはSS／SPで設定します．内部割り込みコードによってコマンドを呼び出します(INT)．

●注意事項

グラフLIOコマンド実行中はハードウェア割り込みが可能な状態となっています．

初期化(GINT)

機　能

グラフLIOの初期化を行います．このコマンドを使用することにより，次のような状態に初期化されます．

・グラフィックチャージャーのリセット．
・画面モードはカラーグラフィックモードに設定されます．
・アクティブ画面は0になります(ページ0のみ描画可能)．
・ディスプレイ画面は1になります(ページ0のみ描画可能)．
・パレットモードは0になります(8色／8色)．

・パレット番号は色表示コード(8色)に対応します.
・フォアグランドカラーはパレットナンバー7に対応します.
・バックグランドカラーはパレットナンバー0に対応します.
・ボーダーカラーはパレットナンバー0に対応します.
・表示モードは0になります(標準カラーモード　640×200).
・表示スイッチは0になります(グラフィック表示有り，普通描画／UV 2の場合は高速描画)

入　力

内部割り込みコード：0A0H

DS：グラフLIOのワークエリアセグメントベース(ユーザーUCW)

ワークエリアはユーザーが確保して下さい．GCOPYコマンドを使用する場合は，DSの相対アドレスから＋1400 Hバイト，使用しない場合は1200 Hバイト確保する必要があります.

SS／SP：ユーザが確保するスタックエリア

コマンド実行のために必要なスタックエリア約128バイトはユーザーが確保して下さい.

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

グラフィック画面に対するモード設定(GSCREEN)

機　能

画面モード，画面スイッチ，アクティブ画面，ディスプレイ画面を設定します．このコマンドを実行することにより，描画領域はアクティブ画面全体が対象となります.

入　力

内部割り込みコード：0A1H

DS：グラフLIOワークエリアのセグメントベース(1200 Hバイト確保する.)

SS／SP：ユーザースタックエリア

BX：パラメータリストの先頭オフセットアドレス(パラメータリストはデータセグメント内に設定)

+0	+1	+2	+3
画面モード	画面スイッチ	アクティブ画面	ディスプレイ画面

+4

↑
BX

パラメータ	画面モード	画面スイッチ＊1
00H	カラーグラフィック	グラフィック表示有
01H	グラフィック	グラフィック表示有(高速)
02H	高解像度ディスプレイ	グラフィック表示無(高速)
03H	高解像度カラーディスプレイ	グラフィック表示無(高速)＊2
FFH	今までのモード	今までのモード

アクティブ画面……00H～0BH(＊3)：アクティブ画面の番号
FFH(省略形)：画面モードに変更がない場合は，今までのアクティブ画面を引き継ぎます．画面モードに変更がある場合は，アクティブ画面を0とします．

ディスプレイ画面……00H～1FH(＊4)：ディスプレイ画面の番号
FFH(省略形)：画面モードに変更がない場合は，今までのディスプレイ画面を引き継ぎます．画面モードに変更がある場合は，ディスプレイ画面を1とします．

＊1：PC-9801UVでは，拡張グラフィックモードを選択した時，常に高速書き込みを行います．そのため，書き込みモードの選択は無意味です．
＊2：PC-9801U／VF／VMでは，画面スイッチに02Hを指定した場合，高速書き込みを行いません．
＊3：PC-9801／Uは00H～05H
＊4：PC-9801／Uは00H～0FH

図2-3-32　パラメータリスト

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00H：正常終了
05H：不正呼び出し(不処理)

画面モード，アクティブ画面，ディスプレイ画面の組み合わせが不適当な場合はエラーとなります．

①画面モード

パラメータ	画面モード名称	分解能	色	画面数(注)	使用装置
00H	カラーグラフィックモード	640×200	カラー	2×2対	標準・専用高解像度
01H	グラフィックモード	640×200	モノクロ	6×2対	同　上
02H	専用高解像度ディスプレイモード	640×400	モノクロ	3×2対	専用高解像度
03H	専用高解像度カラーディスプレイモード	640×400	カラー	1×2対	同　上
FFH	省略，今までの画面モードを引継ぐ				

注：PC-9801／Uの場合は1対

②画面スイッチ

パラメータ	グラフィック画面の表示有無	高速描画有無
00H	グラフィック画面表示　有	普通描画・高速描画　無
01H	グラフィック画面表示　有	高速描画　有
02H	グラフィック画面表示　無	高速描画　有(注)
03H	グラフィック画面表示　無	高速描画　有
FFH	省略，今までの画面スイッチの状態を引継ぐ	

注：PC-9801U／VF／VMでは，画面スイッチに02Hを指定した場合，高速書き込みを行わない．

③アクティブ画面

画面モード	パラメータ指定範囲		G-VRAM使用法
	PC-9801／U	その他	
カラーグラフィックモード	0～1	0～3	G-VRAMを2つに分割して使用
グラフィックモード	0～5	0～11	G-VRAMを6つに分割して使用
専用高解像度ディスプレイモード	0～2	0～5	G-VRAMを3つに分割して使用
専用高解像度カラーディスプレイモード	0	0～1	G-VRAMすべてを使用

④ディスプレイ画面

注：PC-9801／Uについては(　)の部分は適用されません．

●カラーグラフィックモード

パラメータ	表示画面
0，8，(16)	表示しない
1	画面0を表示
2	画面1を表示
(17)	(画面2を表示)
(18)	(画面3を表示)

●専用高解像度カラーディスプレイモード

パラメータ	表示画面
0，8，(16)	表示しない
1	画面0を表示
(17)	(画面1を表示)

●専用高解像度ディスプレイモード

パラメータ	表示画面
0，8，(16)	表示しない
1(17)	画面0(3)を表示
2(18)	画面1(4)を表示
3(19)	画面0(3)と1(4)を合成して表示
4(20)	画面2(5)を表示
5(21)	画面0(3)と2(5)を合成して表示
6(22)	画面1(4)と2(5)を合成して表示
7(23)	画面0(3)，1(4)，2(5)を合成して表示

●グラフィックモード

パラメータ	表示画面
0，(16)	表示しない
1(17)	画面0(6)を表示
2(18)	画面1(7)を表示
3(19)	画面0(6)と1(7)を合成して表示
4(20)	画面2(8)を表示
5(21)	画面0(6)と2(8)を合成して表示
6(22)	画面1(7)と2(8)を合成して表示
7(23)	画面0(6)，1(7)，2(8)を合成
8(24)	表示しない
9(25)	画面3(9)を表示
10(26)	画面4(10)を表示
11(27)	画面3(9)と4(10)を合成して表示
12(28)	画面5(11)を表示
13(29)	画面3(9)と5(11)を合成して表示
14(30)	画面4(10)と5(11)を合成して表示
15(31)	画面3(9)，4(10)，5(11)を合成して表示

表2-3-9　パラメータリスト(詳細)

描画領域の指定

機　能

アクティブ画面内の描画領域(ビューポート)を指定します．また，ビューポート内の塗りつぶしと外枠の描画を行います．このコマンドの実行によって，アクティブ画面への図形描画は次の図のようにビューポート内にのみ反映されます．

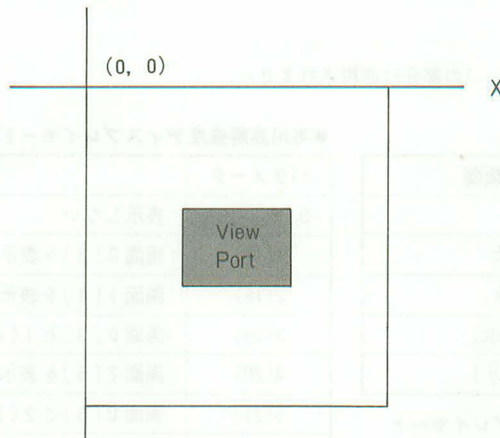


図2-3-33　ビューポート

入　力

内部割り込みコード：0A2H

DS：グラフLIOワークエリア(1200Hバイト)のセグメントベース

SS／SP：スタックエリア(128バイト)を指定

BX：パラメータリストの先頭オフセットアドレス

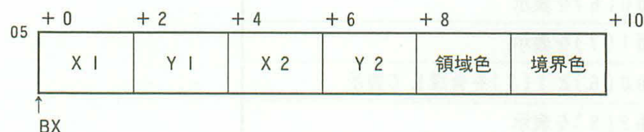


図2-3-34　パラメータリスト

X1：ビューポートの左上X座標

Y1：ビューポートの左上Y座標

X2：ビューポートの右下X座標

Y2：ビューポートの右下Y座標

領域色：ビューポート内を塗りつぶす色を指定……00H～07H：指定パレット番号

FFH：塗りつぶしを行わない

境界色：ビューポート外枠の色指定．指定方法は領域色と同じです．

注意：X1<X2，Y1<Y2が成り立つこと．共にアクティブ画面上の座標であること．

出 力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常

05 H：不正呼び出し

2-3-2-6 グラフ LIO で使用する座標系

①論理座標系

前出のように，-32768 から 32767 までの整数値をとる X，Y で示された座標系です．

②アクテイブ画面領域

0≦X≦639 と 0≦Y≦199 または 399(X，Y は正整数)で示される領域です．

③実際に描画が行われ，表示が可能なのはアクティブ画面領域内です．

2-3-2-7 カラー指定

①パレット番号

図形描画時のカラー指定は，パレット番号と呼ぶ論理的な色で行います．パレット番号は 0 から 7 までの整数です(16 色拡張グラフィックボード使用時は 0 から 15 まで)．パレット番号に絶対色に対応する表示色コードを指定することにより，パレット番号が表す色が定まります．

②表示色コード

表示色コードは表示色を表し，次のように対応し，正整数で指定します．

0-黒　1-青　2-赤　3-紫　4-緑　5-水色　6-黄　7-白

16 色グラフィックボード使用時は次のような色が追加されます．

8 -灰色　9 -暗い青　10-暗い赤　11-暗い紫
12-暗い緑　13-暗い水色　14-暗い黄　15-暗い白

③モノクロモードの場合

パレット番号 0：黒
その他　：白

とみなされて描画が行われます．

背景色等の指定(GCOLOR 1)

機　能

バックグラウンドカラー，ボーダーカラー，フォアグラウンドカラーを指定します．バックグラウンドカラーとは，グラフィック画面の地の色のことで，このコマンドを実行した後にGCLSコマンドによって画面を消去すると，この色によって画面が塗り変えられます．また，以後にGPSETコマンドを色指定なしで実行するとこのコマンドで指定された色が使われます．ボーダーカラーとはグラフLIOが制御可能なディスプレイの外側の色のことで，高解像度ディスプレイ使用時には無意味となります．フォアグランドカラーとは，図形描画においてパレット番号が省略されたときにデフォルトとして使用される色です．

入　力

内部割り込みコード：0A3H

DS：グラフLIOワークエリア(1200Hバイト)のセグメントベース

SS／SP：グラフLIOスタックエリア(128バイト)

BX：パラメータリストの先頭オフセットアドレス

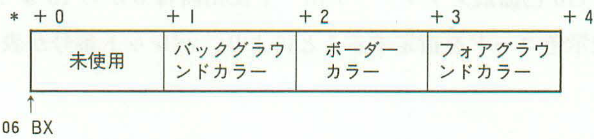


図2-3-35　パラメータリスト

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00H：正常終了

パレット番号と表示色コードの対応(GCOLOR 2)

機　能

パレット番号と表示色コードを対応させ(00H～07H)，パレット番号で表す色コードを定義します．

入　力

内部割り込みコード：0A4H

DS，SS／SP，BXの扱いは今までと同じです．

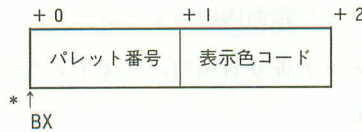


図 2-3-36　パラメータリスト

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

描画領域の塗りつぶし(GCLS)

機　能

アクティブ画面内の描画領域をバックグラウンドカラーで塗りつぶします．

入　力

内部割り込みコード：0 A 5 H

DS，SS，SP の扱いは今までと同じです．

パラメータリストは不要です．

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

ドットの書き込み(GPSET)

機　能

指定された座標に，指定された色のドットを書き込みます．色はパレット番号で指定します．

入　力

内部割り込みコード：0 A 6 H

DS，SS，SP，BX の扱いは今までと同じです．

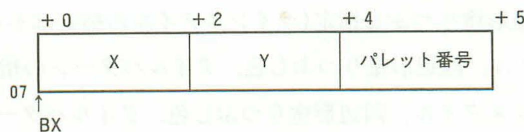


図 2-3-37　パラメータリスト

X／Y：ドットを打つ座標(アクティブ画面内のビューポート)

AH：動作モード……01 H：パレット番号省略時，フォアグラウンドカラーのパレット番号を使用

02 H：パレット番号省略時，バックグラウンドカラーのパレット番号を使用

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

直線／矩形の描画(GLINE)

機　能

指定された2点を結ぶ直線，またはこの2点を対角線とする矩形を描画します．矩形内の塗りつぶしも可能です(PC 9801では不可．E／M／F／VF／VM／U／UVで可能です)．

入　力

内部割り込みコード：0 A 7 H

DS，SS，SP，BXの扱いは今までと同様です．

タイルパターン格納域を設定します．

+0	+2	+4	+6	+8	+9	+10	+11	+12	+13	+14	+16 (+18)
X1	Y1	X2	Y2	パレット番号1	描画コード	スイッチ	パレット番号2	ラインスタイルHi	TPL	TPO	TPB

↑BX

図2-3-38　パラメータリスト(PC-9801 U 2／VM／VFの場合)

X 1／Y 1：描画開始点の座標

X 2／Y 2：描画終了点の座標

パレット番号1：四辺形を描画する色コード(パレット番号)

描画コード……00 H：直線描画指定

01 H：矩形描画指定

02 H：矩形塗りつぶし指定(ラインスタイルの指定は不可)

スイッチ：ラインスタイル，四辺形塗りつぶし色，タイルパターンの指定スイッチ

00 H：ラインスタイル，四辺形塗りつぶし色，タイルパターン指定無し

01 H：ラインスタイルまたはパレット番号2の指定あり

02 H：タイルパターン指定あり

(描画コードが 02 H の場合，02 H を指定すると四辺形内部は描画色で塗りつぶされます)

パレット番号 2：四辺形内部の塗りつぶし色(描画コードが 02 H の時のみ有効)

ラインスタイル Hi：直線または矩形を描く線のパターンを 16 ビットのドットで示す(指定が無いと FFFFH とみなす)

タイルパターン格納域の形式については「GPAINT 2」を参照してください．

TPL(タイルパターン長)：00 H～FFH，描画コードが 02 H の時に有効

TPO(タイルパターンオフセット)：オフセットアドレス　000 H～FFFFH

TPB(タイルパターンベース)：セグメントアドレス　000 H～FFFFH

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

<ラインスタイルの表現>

BX+11(Lo)，BX+12(Hi)で表現されるビット位置と，ディスプレイ上に表現される(VRAM上)ドット位置との関係は下の図のようになります．ビットが ON ならばそこに対応してドットが打たれます．指定がないと FFFFH とみなされます．

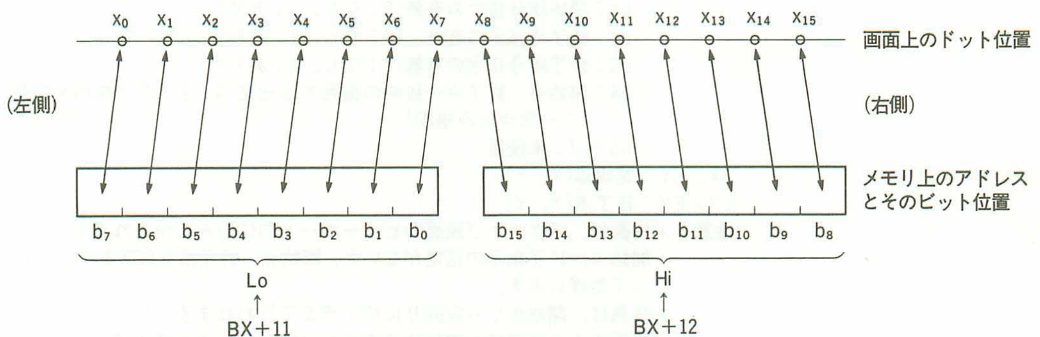


図 2-3-39　ラインスタイル対応表

注　意

- 描画はアクティブ画面領域内にのみ，反映されます．
- ラインスタイルの指定がない時は，ラインスタイルとして FFFFH が指定されたものとみなされます．
- 描画コードに 02 H を指定した場合，ラインスタイルを指定することはできません．
- 描画コードに 02 H を指定し，パレット番号 2 もタイルパターンも指定しない場合，四辺形の内部は四辺形の描画色で塗りつぶされます．
- パレット番号 2 およびタイルパターンの指定は描画コードが 02 H の時のみ，可能です．
- タイルパターン格納域の形式は「GPAINT 2」を参照して下さい．

円・楕円の描画(GCIRCLE)

機　能

指定された中心点，X方向半径，Y方向半径をもとに円または楕円を描画します．開始点，終了点を指定することにより，円弧・扇形を描画することも可能です．PC-9801以外の機種(E／F／M／VM／VF／U／UV)では，描画した図形の内部を塗りつぶすこともできます．

入　力

内部割り込みコード：0A8H

DS，SS，SP，BXの扱いは以前と同様です．

タイルパターン格納域を設定します．

<PC-9801>

+0	+2	+4	+6	+8	+9	+10	+12	+14	+16
CX	CY	RX	RY	パレット番号	フラグ	SX	SY	EX	EY

↑
BX　(+18)

CX／CY：中心点(X，Y)
RX：X方向半径
RY：Y方向半径
パレット番号：楕円の円周を描画する表示色……00H～07H：パレット番号
FFH：フォアグラウンドカラー
フラグ……b0：開始点指示の有無　(0：なし，1：あり)
b1：開始線分指示の有無(0：なし，1：あり)
b2：終了点指示の有無　(0：なし，1：あり)
b3：終了線分指示の有無(0：なし，1：あり)
b4：開始点・終了点一致時の描画方法指定(0：すべての楕円を描画，1：一致点のみ描画)
b5～b7：未使用
SX／SY：開始点(X，Y)
EX／EY：終了点(X，Y)

注意：
- 描画は，アクティブ画面のビューポート内にのみ反映されます．
- 開始点，終了指示の指定がないと，開始点，終了点を(CX+RX，CY)として処理します．
- 描画は，開始点から左回りに終了点まで行われます．
- 描画する点の座標が整数値で表せない場合は，その時点でエラーリターンとします．
- 開始点，終了点は，描画する楕円上の点でなければなりません．理論的に求めた値を四捨五入した値が座標になります．
- CX，CY，RX，RY，SX，SY，EX，EYは整数値です．

<PC-9801E／F／M／U／UV／VF／VM>

+0	+2	+4	+6	+8	+9	+10	+12	+14	+16
CX	CY	RX	RY	パレット番号	フラグ	SX	SY	EX	EY

↑
BX

+18	+19	+21 (+23)
パレット番号2	タイルパターンオフセット	タイルパターンベース
タイルパターン	Lo / Hi	
	オフセットアドレス	セグメントベースアドレス

── <タイルパターン格納域> ──

フラグ……b0～b4：PC-9801と同じ
b5：塗りつぶし指示の有無(0：なし，1：塗りつぶす)
b6：タイルパターン指示の有無(0：なし，1：あり)
パレット番号2……00H～07H：指定パレット番号
FFH：円または扇形描画と同じ色
タイルパターン(格納域長)：00H～FFH
注意：円弧を描くように開始・終了座標を指定し，塗りつぶし指示ありと指定した場合には開始・終了線分指示(ビット1，3)の指定にかかわらず扇形を描画し，その内部を塗りつぶします．
全円を描画する場合には（開始，終了座標の指定があり，それらが互いに等しく，描画方向指定が0の場合も含む)，開始・終了線分の描画は指示に従います．
特に説明のない部分はPC-9801と同様です．

図2-3-40 パラメータリスト

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了
06 H：演算オーバーフロー

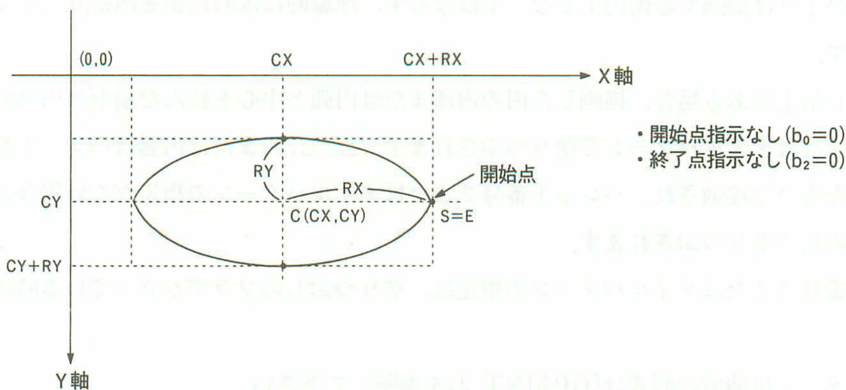


図2-3-41 楕円(描画開始点，終了点の指定なし)

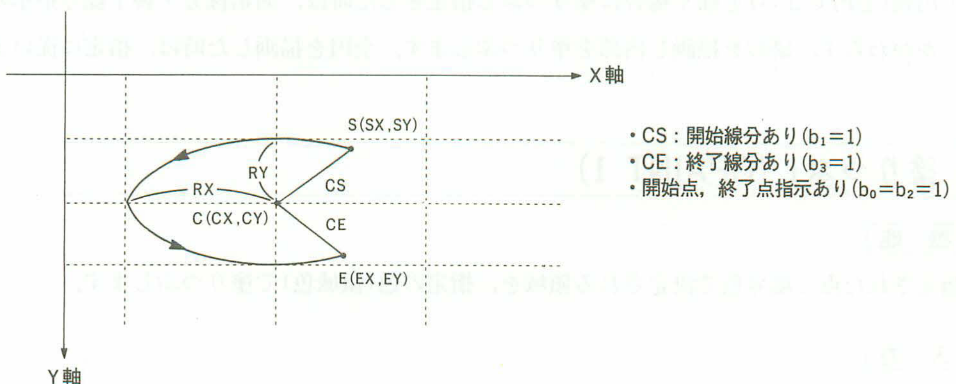


図2-3-42 開始点S，終了点E，開始線分CS，終了線分CE

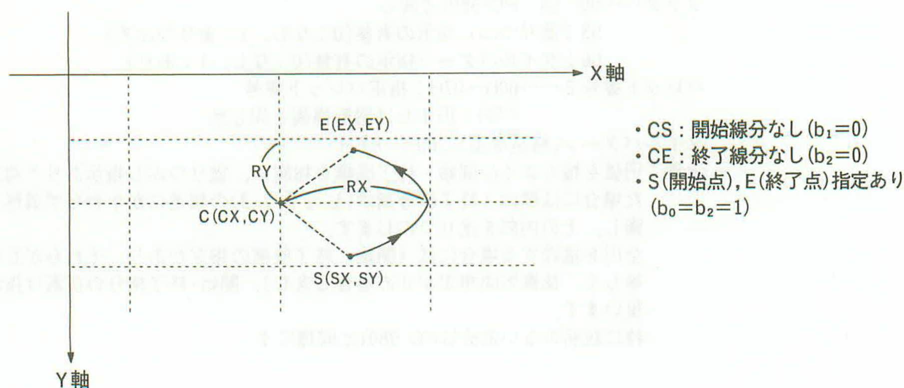


図2-3-43 孤の描画

注 意

・描画はアクティブ画面内のビューポート内にのみ，反映されます．
・開始点，終了点の指定が無い場合は，終了点を(CX+RX，CY)とみなします．
・描画は開始点から終了点まで左回りに行われます．
・描画する点の座標が整数値で表せない場合は，その時点でエラーリターンします．
・開始点，終了点は描画する楕円上でなくてはならず，理論的に求めた値を四捨五入して座標を決定します．
・塗りつぶし指定がある場合，描画した円の内部または円弧と中心を結んだ扇形の内部がパレット番号2またはタイルパターンで塗りつぶされます．ただし，円または円弧はパレット番号1で指定されたもので描画され，パレット番号2またはタイルパターンの指定がない場合はパレット番号1の色で塗りつぶされます．
・パレット番号2またはタイルパターンの指定は，塗りつぶしのフラグが立っている時のみ，有効です．
・タイルパターン格納域の形式は「GPAINT 2」を参照して下さい．
・CX，CY，RX，RY，SX，SY，EX，EYは整数値です．
・円弧(全円でない)を描く場合に塗りつぶし指定をした時は，開始線分・終了線分指示の指定にかかわらず，扇形を描画し内部を塗りつぶします．全円を描画した時は，指定に従います．

塗りつぶし(GPAINT 1)

機 能

指定された点と境界色で決定される領域を，指定の色(領域色)で塗りつぶします．

入 力

内部割り込みコード：0A9H

DS，SS，SP，BXの扱いは以前と同じです．

+0	+2	+4	+5	+6	+8	+10
X	Y	領域色	境界色	ワークエリア最終	ワークエリア先頭	

↑
BX

図 2-3-44 パラメータリスト

X／Y：塗りつぶしの開始点(アクティブ画面内のビューポート)

領域色：領域を塗りつぶす色(パレット番号)

境界色：境界の表示色(パレット番号)

ワークエリア：ペイントのための作業域(16 バイト以上)

(ワークエリア最終オフセットアドレス)-(先頭アドレス)＞16 バイト

ワークエリアはデータセグメント内になければなりません．また，このエリアは他の目的のために使用してはいけません．

注 意

ワークエリアが実行中に足らなくなると，使いきった時点で中断しエラーリターンとなります．

出 力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常

05 H：不正呼び出し

07 H：ワークエリア不足

タイルパターンによる塗りつぶし(GPAINT 2)

機 能

指定された点と境界色で決定される領域を，指定されたタイルパターンで塗りつぶします．

入 力

内部割り込みコード：0 AAH

DS，SS，SP，BX の扱いは今までと同じです．

+0	+2		+5	+6	+8	+10				+16	+18
X	Y		タイルパターン長	タイルパターンオフセットアドレス	タイルパターンセグメントベース	境界色				ワーク域最終アドレス	ワーク域先頭アドレス

↑
BX

図 2-3-45 パラメータリスト

X／Y：塗りつぶしの開始点

タイルパターン長：タイルパターン格納域の大きさ(バイト単位)

01H～FFH(モノクロ)

03H～FFH(カラー)

タイルパターンオフセットアドレス：格納域のオフセットアドレス

(0000H～FFFFH)

タイルパターンセグメントベース：格納域のセグメントアドレス

(0000H～FFFFH)

境界色：パレット番号

ワークエリアの扱いは「GPAINT 1」と同じです．

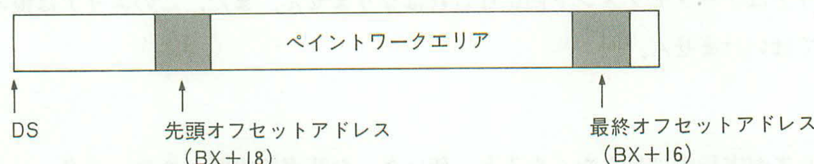


図2-3-46　パラメータリスト(ペイントのためのワークエリアのアドレス指定)

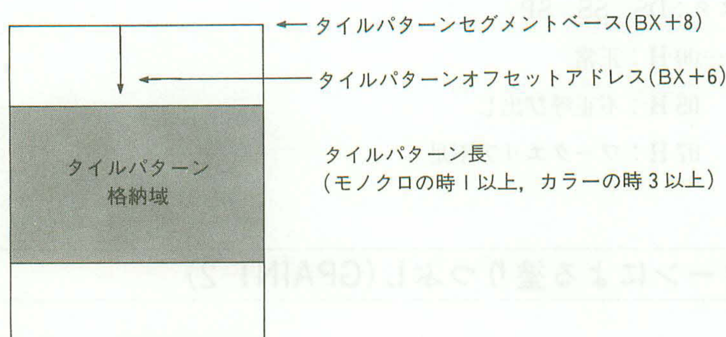


図2-3-47　パラメータリスト(タイルパターン格納域のアドレス指定)

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00H：正常

05H：不正呼び出し

07H：ワークエリア不足

タイルパターン格納域に格納するタイルパターンの形式は次のようになります．

①画面モードがカラーの場合

タイルパターンは，横8ドットを一組に，縦方向に必要分だけ，横8ドットごとに定義したド

ットの集まりから成り立ち，この基本パターンによって，指定されたビューポートの指定された開始点から塗りつぶしていきます．実際の描画は，埋められたパターンの中の指定された領域について反映されます．カラーの場合，各ドットごとにパレット番号によって表示色が定義されます．各ドットに対応するパレット番号は次のようにして定義されます．画面上の横８ドットが，次の様に３バイト(16色の場合は４バイト)のビットごとに対応し，このビット情報によって，対応するドットのパレット番号を表します．

Pi←(b 4, ib 3, ib 2, ib 1, i)＝b 4, i・2^3＋b 3, i・2^2＋b 2, i・2＋b 1, i

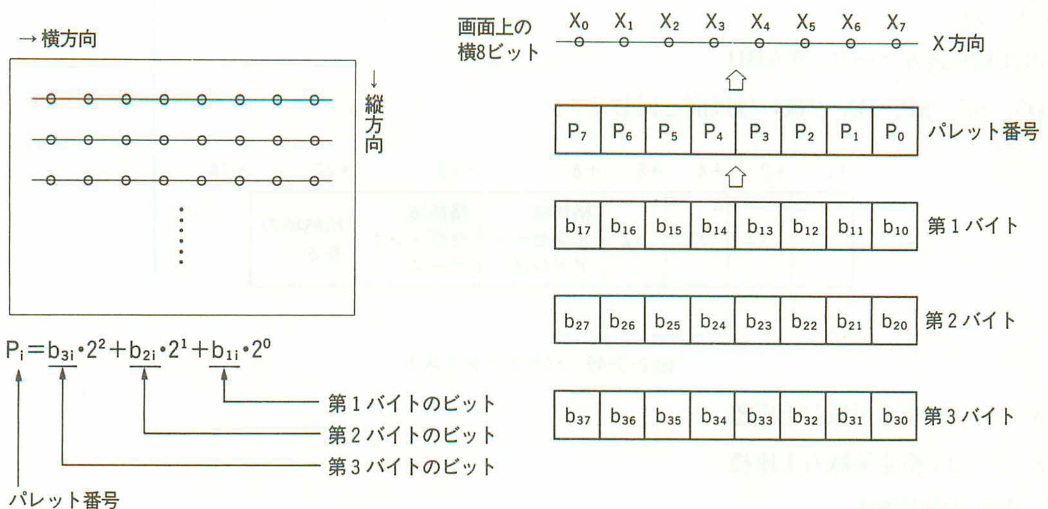


図 2-3-48 パレット番号

②画面モードがモノクロの場合

タイルパターンは横８ドットの白・黒表示を一組に，カラーの時と同様に定義されたドットの集まりから成立ち，ビットが１ならば白，０ならば黒という対応で表示されます．

③タイルパターン格納域のデータ表現

・カラーモード

X^1_{1n}　X^0_{1n}　X^1_{2n}　X^0_{2n}　X^1_{3n}　X^0_{3n}

第１バイト　第２バイト　第３バイト

＜第ｎ列の横８ドットを表す＞

・モノクロモード

X^1_n　X^0_n

第ｎ列　……それぞれの横８ドットを表す

④タイルパターンの描画位置

タイルを張り詰める位置は，ビューポートの基点(左上)から始まります．

描画情報の格納(GGET)

機　能

指定された領域の描画情報を指定された格納域へ格納します．

入　力

内部割り込みコード：0ABH

DS，SS，SP，BXの扱いは以前と同じです．

+0	+2	+4	+6	+8	+10	+12
X1	Y1	X2	Y2	格納域オフセットアドレス	格納域セグメントベース	格納域の長さ

+14

↑ BX（+0）

図2-3-49　パラメータリスト

X1／Y1：指定領域左上座標

X2／Y2：指定領域右上座標

・座標の指定条件

(X1, Y1), (X2, Y2)は共にアクティブ画面上のビューポート内にあり，かつX2≧X1, Y2≧Y1であること(X，Yは整数値)

・格納域の指定条件

¥は整数の割り算の商(余り切捨て)，・は乗算

<画面モードカラーの場合>

格納域の長さ≧((X2-X1+8)¥8)・(Y2-Y1+1)・3+4

<画面モードモノクロの場合>

格納域の長さ≧((X2-X1+8)¥8)・(Y2-Y1+1)+4

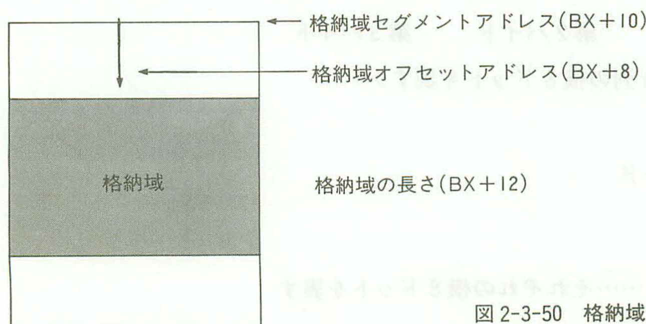


図2-3-50　格納域

出 力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

05 H：不正呼び出し

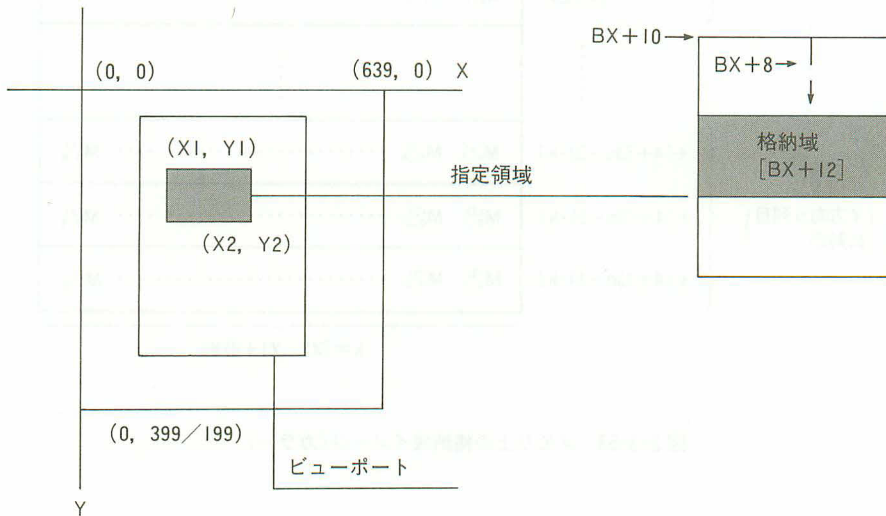


図 2-3-51 指定領域と格納域

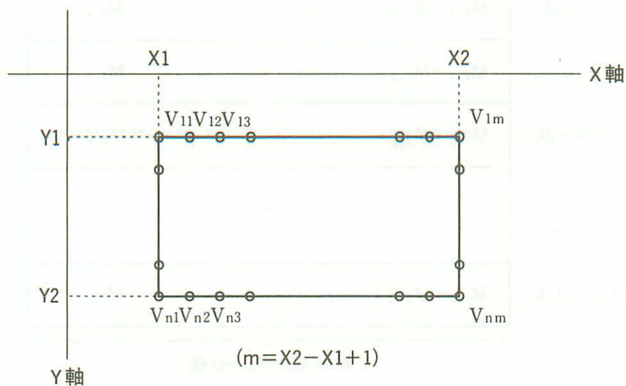


図 2-3-52 格納域の形式(カラー・モノクロの画面イメージ)

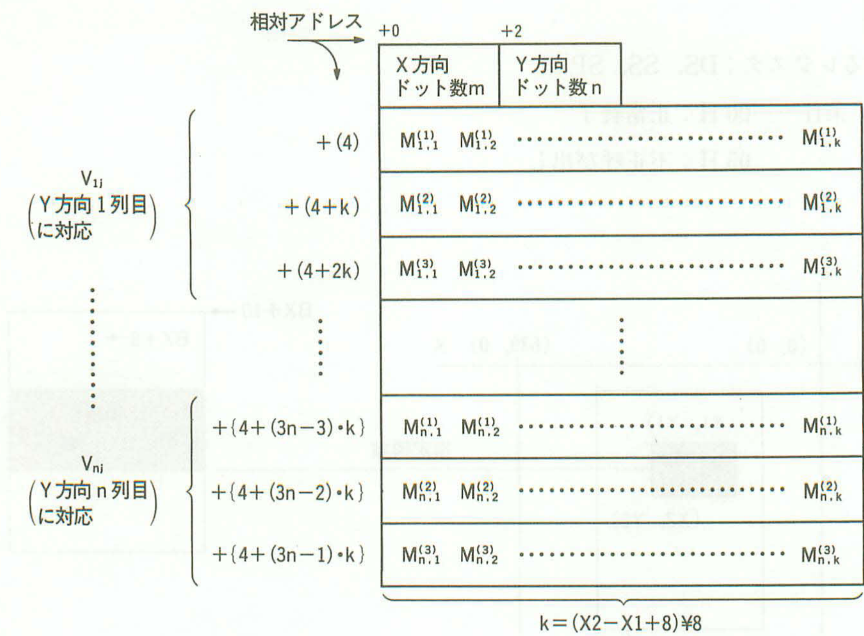


図2-3-53　メモリ上の格納域イメージ(カラー)

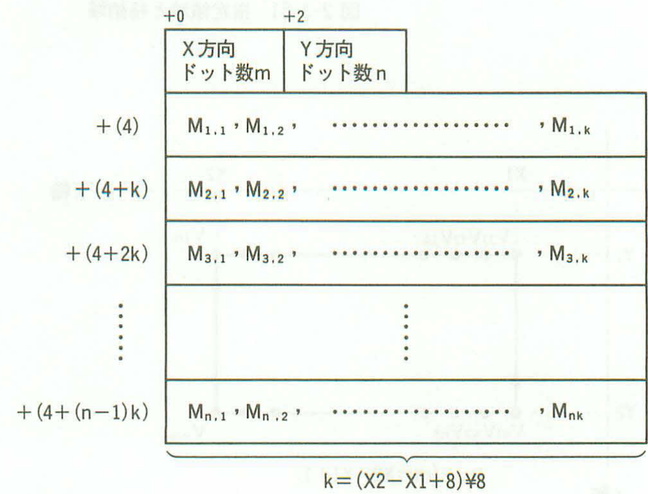


図2-3-54　メモリ上の格納域イメージ(モノクロ)

画面情報を格納域から領域へ戻す(GPUT 1)

機　能

指定された格納域内の描画情報を，指定された領域上に設定します．

入　力

内部割り込みコード：0 ACH

DS，SS，SP，BX の扱いは以前と同じです．

+0	+2	+4	+6	+8	+10	+11	+12	+13
X	Y	格納域オフセットアドレス	格納域セグメントベース	格納域長さ	描画モード	カラースイッチ	フォアグラウンドカラー	バックグラウンドカラー

(+14)

↑BX

図 2-3-55　パラメータリスト

X／Y：描画座標(アクテイブ画面のビューポート内)

格納域：格納域のオフセットとアドレスとセグメントベース，長さ

描画モード：次表を参照

カラースイッチ……00 H：フォア，バックグランドカラー指定なし，画面モードは現在のモード

01 H：フォア，バックグラウンドカラー指定あり，画面モードはモノクロモード

フォア／バックグラウンドカラー：モノクロモードで格納されている描画パターンの白(1)，黒(0)のドットを表示するときのパレット番号(白はフォア，黒はバックグラウンドカラー)

描画モード……指定領域上のパターン：A 0

格納域の描画パターン：B

格納域の描画パターンで操作した指定領域上の描画パターン：AN

とすると，A 0 に対し，B で OP 操作を行った結果が AN となります．描画モード(00 H～04 H)の OP を次に示します．

描画モード	操　作	機　能
00H	B　　→AN	置　換
01H	→AN	
02H	A0＋B→AN	論理和
03H	A0×B→AN	論理積
04H	A0－B→AN	排他的論理和

表 2-3-10　描画モード

注　意

・格納域左上点(X, Y), 右下点(X+X方向ドット-1, Y+Y方向ドット-1)は, アクティブ画面の描画領域内になければなりません. さもなくば, 不処理となり, エラーリターンします.
・カラーモードにおけるそれぞれの操作は各パレット番号を表現する3(4)ビットのビット列に対して論理演算を行います.
・フォアグラウンドカラー, およびバックグラウンドカラーの指定は画面モードがカラーの場合にのみ意味を持ちます.

出　力

保証されるレジスタ：DS, SS, SP

AH：終了条件……00 H：正常

05 H：不正呼び出し

日本語の描画(GPUT 2)

機　能

指定された日本語(JISコード)を, 指定された領域上に描画します.

入　力

内部割り込みコード：0 ADH

DS, SS, SP, BXの扱いは以前と同じです.

+0	+2	+4	+6	+7	+8	+9
X	Y	日本語コード	描画モード	カラースイッチ	フォアグラウンドカラー	バックグラウンドカラー

↑
BX

図2-3-56　パラメータリスト

X, Y：描画領域左上座標

日本語コード：JISコード(0000 H～FFFFH)

描画モードとカラースイッチは「GPUT 1」を参照してください.

フォアグラウンドカラー：描画する文字のパレット番号

バックグラウンドカラー：描画する領域の文字パターン以外の部分

・描画領域

指定の日本語が,

全角の場合　：(X，Y)～(X+15，Y+15)
半角の場合　：(X，Y)～(X+7，Y+15)
1/4角の場合：(X，Y)～(X+7，Y+7)

上記領域がアクティブ領域内になければエラーリターンとなります.

出　力

保証されるレジスタ：DS，SS，SP
AH：終了条件……00 H：正常
　　　　　　　　05 H：不正呼び出し

描画画面の移動(GROLL)

機　能

アクティブ画面全体の描画情報を指定ドット数分上下左右に移動します.

入　力

内部割り込みコード：0 AEH
DS，SS，SP，BX の扱いは以前と同じです.

+0	+2	+4 (～+5)
上下ドット数 −399～399	左右ドット数 −639～639	クリアフラグ 0／1

↑
BX

図 2-3-57　パラメータリスト

クリアフラグ……00 H：移動後の残りの領域をパレット 0 に指定
　　　　　　　　01 H：残り領域をバックグラウンドカラーに指定

出　力

保証されるレジスタ：DS，SS，SP
AH：終了条件……00 H：正常終了
　　　　　　　　05 H：不正呼び出し

注　意

・描画情報移動後の残り領域にはクリアフラグに従ってパレット番号 0 かまたは，バックグラウンドカラーが設定されます.
・表示モードが標準 CRT の場合，上下ドット数の指定は-199～199 の範囲に限られます.

・上下ドット数が正の場合は上方向，負の場合は下方向へ，左右ドットが正の場合は左方向へ，負の場合は右方向へスクロールします．
・左右方向へスクロールする場合，実際に移動するドット数は指定されたドット数の絶対値以下でもっとも近い8の倍数分です．

ドットに対応するパレット番号の取得(GPOINT 2)

機　能

指定された座標のドットのパレット番号を取得します．

入　力

内部割り込みコード：0 AFH

DS，SS，SP，BXの扱いは以前と同じです．

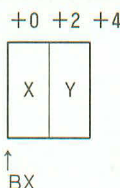


図2-3-58　パラメータリスト

X／Y：パレット番号を求めるドットの座標

ES：DS

出　力

保証されるレジスタ：DS，SS，SP

AH：終了条件……00 H：正常終了

AL：ドットのパレット番号……FFH：指定座標がアクティブ画面のビューポートにない

00 H～07 H：画面モードがカラーの場合，指定座標のパレット番号を示します．

00 H／01 H：画面モードがモノクロの場合，00 H-黒，01 H-白を示します．

表示画面のドット情報を格納域へ設定する(GCOPY)

機 能

現在のディスプレイ画面の指定領域のドット状態を，指定された格納域へ設定します．ここでいう「ドット状態」とは，

画面モードがカラーの場合　：表示中のドットのパレット番号が0であれば0,それ以外ならば1とします．

画面モードがモノクロの場合：画面の合成も含めて，表示中のドットが黒ならば0,白ならば1とします．

入 力

内部割り込みコード：0 CEH

DS：グラフ LIO ワークエリアセグメントベース

グラフ LIO ワークエリアは 1400 H バイト確保します．

SS／SP：グラフ LIO スタックエリア(128 バイト)

AX：指定領域左上 X 座標(0000 H～027 FH)……X

BX：指定領域左上 Y 座標(0000 H～018 FH)……Y

CL：指定領域 X 方向ドット数(00 H～FFH)……dx

CH：指定領域 Y 方向ドット数(02 H／82 H，04 H／84 H，08 H)…dy

DI：格納域のオフセットアドレス(0000 H～FFFFH)

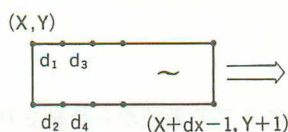
ES：格納域のセグメントベース(0000 H～FFFFH)

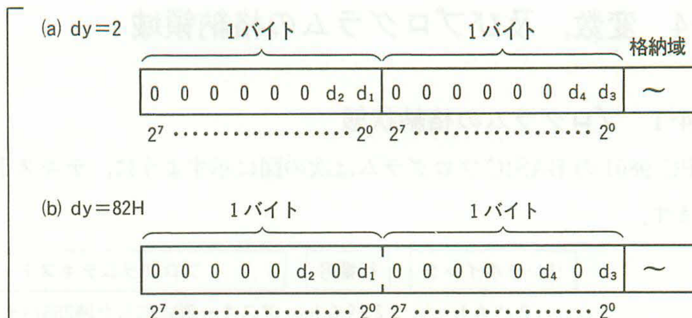
<X，Y，dx，dy についての注意>

① X，dx は 8 の倍数であり，かつ X＋dx-1≦027 FH を満たすこと．

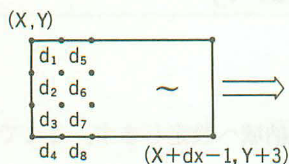
② Y，dy

標準モードの場合，Y≦00 C 7 H　かつ　Y＋dy-1≦00 C 7 H を満たすこと．また，カラーモードの場合，Y≦018 FH　かつ　Y＋dy-1≦018 FH を満たすこと．

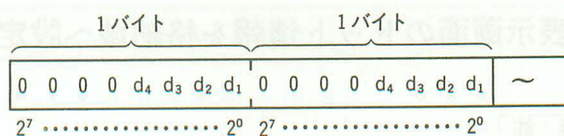




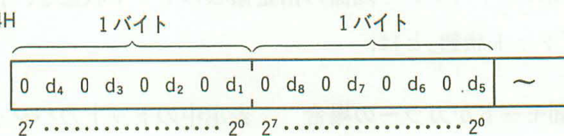
② DY=4, または84Hの場合



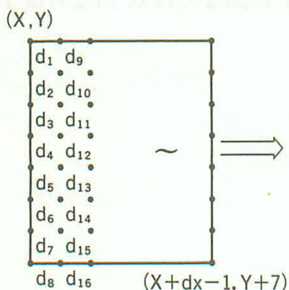
(a) dy=4



(b) dy=84H



③ DY=8



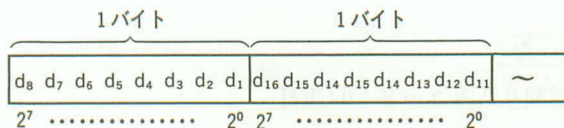


図2-3-59　格納域対応表

<表示ビットと格納領域ビットとの関係>

①カラーモード

ドットのパレット番号が0ならば，ビット表現は0，ドットが0以外ならばビットは1となります．

②モノクロモード

ドットが黒ならばビットは0，ドットが白ならばビットは1となります．

出　力

保証されるレジスタ：DS, SS, SP

AH：不定

2-4　変数，及びプログラムの格納領域

2-4-1　プログラムの格納状態

PC-9801のBASICプログラムは次の図に示すように，テキストエリアの先頭から格納されています．

リンクポインタ	行番号	プログラムテキスト	0 0
2バイト	2バイト	アスキー形式にした時255バイト以内	1バイト

図2-4-1　BASICプログラムの格納形式

・リンクポインタ

PC-9801ではリンクポインタは88シリーズや80シリーズと違って，アドレスではなく，その行の占めるバイト数が入っています．プログラムの終わりではリンクポインタの値は0となります．

・プログラムテキスト

BASICのプログラムが中間言語に変換されて格納されています．テキストのTOP, ENDはセグメント60Hのオフセット6A4H~6A7Hに格納されています．ここで得られたアドレスをダンプして直接格納されたプログラムを見ることができます．

2-4-2 中間言語

2-4-2-1 中間言語コード(00H~7FH)

中間言語コードの00H~7FHは,数値や行番号,変数名に使われ,今までのBASIC(80, 88系)と違うのは01~09がスペースの数になっている点で，BASICプログラムで段付けや空白部分のメモリの節約ができることです．

00H~7FHまでの中間言語コードと実際のプログラムとの対応を下の図に示します．

中間言語		意味	備考
0	REM エンドマーク	そこからあとはREMと同じあつかい(文の途中)，または，行の終わり	
1~9	スペース	スペース1~9コ	
0A	LF	Line Feed	CTRL + J で入力する
0B	&O	以下の2バイトは8進数	0B 9C 02=&O1234
0C	&H	以下の2バイトは16進数	0C 34 12=&H1234
0D	アドレス	以下の2バイトは飛び先オフセットアドレス	GOTO, GOSUB, THEN, ELSE, RESTORE等の後に続きます．
0E	行番号	以下の2バイトは飛び先行番号	
0F	整数	以下の1バイトは，10~255の整数	0F 50=80(10)
10~19	整数	1桁の整数 (10→0, 11→1, ……, 19→9)	
1A		使われていない。Syntax errorになる．	
1B		漢字のシフトイン，シフトアウト	
1C	整数	以下の2バイトは，整数	1C D2 04=1234
1D	単精度	以下の4バイトは，単精度定数	1D EB C0 1D 81=1.2345
1F	倍精度	以下の8バイトは，倍精度定数	
20 ∫ 7F	文字	キャラクタコードに対応する文字 (変数名やラベル名など)	DATA文やREM文，クォテーションの中以外では，アルファベットの小文字は大文字に変換されるため使われません．上記文の中のコードはインタプリタは中間コードとしては実行しません．

表2-4-1 中間コード00H~7FH

2-4-2-2　中間言語コード(80 H～FFH)

中間コードの80 H～FFHまでは1バイトまたは2バイトでN 88-BASIC(86)のキーワードを示します．

N 88-BASICやN-BASICのものと中間コードが異なるので，バイナリでセーブされたプログラムはそのままでは動作しません．N 88-BASICのプログラムでアスキーセーブされたものならN 88-BASIC(86)で動作させる事もできます．

＜1バイトで表わされる中間コード＞

80 (90)	AUTO	B0 (80)	LINE	E0 (80)	WHILE
81 (80)	BSAVE	B1 (80)	LOAD	E1 (80)	WEND
82 (80)	BLOAD	B2 (80)	LSET	E2 (80)	WRITE
83 (80)	BEEP	B3 (80)	LFILES	E3 (90)	LIST
84 (80)	CONSOLE	B4 (80)	MOTOR	E4 (80)	SEG
85 (80)	COPY	B5 (80)	MERGE	E5 (80)	SET
86 (80)	CLOSE	B6 (80)	MON	E6 (80)	KINPUT
87 (80)	CONT	B7 (80)	NEXT	E7 (80)	SRQ
88 (80)	CLEAR	B8 (80)	NAME	E8 (80)	CMD
89 (80)	CALL	B9 (80)	NEW	E9 (80)	IRESET
8A (80)	COMMON	BA (80)	NOT	EA (80)	ISET
8B (80)	CHAIN	BB (80)	OPEN	EB (80)	POLL
8C (80)	COM	BC (80)	OUT	EC (80)	RBYTE
8D (80)	CIRCLE	BD (80)	ON	ED (80)	WBYTE
8E (80)	COLOR	BE (80)	OPTION	EE (80)	KPLOAD*
8F (80)	CLS	BF (80)	OFF	EF (00)	
90 (90)	DELETE	C0 (00)	?	F0 (00)	>
91 (A0)	DATA	C1 (80)	PUT	F1 (00)	=
92 (80)	DIM	C2 (80)	POKE	F2 (00)	<
93 (80)	DEFSTR	C3 (80)	PSET	F3 (00)	+
94 (80)	DEFINT	C4 (80)	PRESET	F4 (00)	−
95 (80)	DEFSNG	C5 (80)	PAINT	F5 (00)	*
96 (80)	DEFDBL	C6 (90)	RETURN	F6 (00)	/
97 (00)	DSKO$	C7 (80)	READ	F7 (00)	∧
98 (80)	DEF	C8 (90)	RUN	F8 (80)	AND
99 (D0)	ELSE	C9 (90)	RESTORE	F9 (80)	OR
9A (80)	END	CA (00)		FA (80)	XOR
9B (80)	ERASE	CB (90)	RESUME	FB (80)	EQV
9C (90)	EDIT	CC (80)	RSET	FC (80)	IMP
9D (80)	ERROR	CD (90)	RENUM	FD (80)	MOD
9E (80)	FOR	CE (80)	RANDOMIZE	FE (00)	¥
9F (80)	FIELD	CF (80)	ROLL	FF (80)	REM
A0 (80)	FILES	D0 (80)	SCREEN		
A1 (00)	FN	D1 (80)	STOP		
A2 (80)	DRAW*	D2 (80)	SWAP		
A3 (90)	GO TO	D3 (80)	SAVE		
A4 (90)	GOSUB	D4 (80)	SPC		
A5 (80)	GET	D5 (80)	STEP		
A6 (80)	HELP	D6 (90)	THEN		
A7 (80)	INPUT	D7 (80)	TRON		
A8 (80)	IF	D8 (80)	TROFF		
A9 (80)	KEY	D9 (80)	TAB		
AA (80)	KILL	DA (80)	TO		
AB (80)	KANJI	DB (80)	TERM		
AC (80)	LOCATE	DC (80)	USING		
AD (00)	L?	DD (00)	USR		
AE (90)	LLIST	DE (80)	WIDTH		
AF (80)	LET	DF (80)	WAIT		

注）＊印はPC-9801F·Eで追加されたもの

注）
REMの中間コードはFFになっていますが，実際にはこれが使われず，

```
00  52  45  40
NL  R   E   M
```

の形で格納されます．
文の途中で，中間コード0があらわれるとインタプリタはそこからあとはREM文として処理します．

アスキーセーブとは，SAVE"ファイル名"，A のように，A オプションを使用して書き込むことをいいます．PC-9800 シリーズの BASIC で動作が可能なのは USR，POKE，PEEK などを使っていないものに限られます．ただし，これらのものでも内容を PC-9801 に合わせて書き換えれば動作は可能です．

以下に中間言語コードと N 88-BASIC(86)のキーワードの対応を示します．

＜2 バイトで表わされる中間コード＞

FF 80(00)	DATE$	FF B0(00)	MKS$	FF E0(00)
FF 81(00)	MID$	FF B1(00)	MKD$	FF E1(00)
FF 82(80)	POINT	FF B2(80)	MAP	FF E2(00)
FF 83(80)	PEN	FF B3(00)	OCT$	FF E3(00)
FF 84(00)	TIME$	FF B4(80)	POS	FF E4(00)
FF 85(80)	VIEW	FF B5(80)	PEEK	FF E5(00)
FF 86(80)	WINDOW	FF B6(00)	RIGHT$	FF E6(00)
FF 87(00)		FF B7(80)	RND	FF E7(00)
FF 88(00)		FF B8(80)	SEARCH	FF E8(00)
FF 89(00)		FF B9(80)	SGN	FF E9(00)
FF 8A(00)		FF BA(80)	SQR	FF EA(00)
FF 8B(00)		FF BB(80)	SIN	FF EB(00)
FF 8C(00)		FF BC(00)	STR$	FF EC(00)
FF 8D(00)		FF BD(00)	STRING$	FF ED(00)
FF 8E(00)		FF BE(00)	SPACE$	FF EE(00)
FF 8F(00)		FF BF(80)	TAN	FF EF(00)
FF 90(80)	ABS	FF C0(80)	VAL	FF F0(00)
FF 91(80)	ATN	FF C1(00)	DSKI$	FF F1(00)
FF 92(80)	ASC	FF C2(80)	FRE	FF F2(00)
FF 93(00)	ATTR$	FF C3(80)	VARPTR	FF F3(00)
FF 94(80)	CSRLIN	FF C4(00)	INPUT$	FF F4(00)
FF 95(80)	CINT	FF C5(00)	JIS$	FF F5(00)
FF 96(80)	CSNG	FF C6(00)	KNJ$	FF F6(00)
FF 97(80)	CDBL	FF C7(80)	KTYPE	FF F7(00)
FF 98(80)	CVI	FF C8(80)	KLEN	FF F8(00)
FF 99(80)	CVS	FF C9(00)	KMID$	FF F9(00)
FF 9A(80)	CVD	FF CA(00)	KEXT$	FF FA(00)
FF 9B(80)	COS	FF CB(80)	KINSTR	FF FB(00)
FF 9C(00)	CHR$	FF CC(00)	AKCNV$	FF FC(00)
FF 9D(80)	DSKF	FF CD(00)	KACNV$	FF FD(00)
FF 9E(90)	ERL	FF CE(80)	IEEE	FF FE(00)
FF 9F(80)	ERR	FF CF(80)	STATUS	FF FF(00)
FF A0(80)	EXP	FF D0(00)		
FF A1(80)	EOF	FF D1(00)		
FF A2(80)	FIX	FF D2(00)		
FF A3(80)	FPOS	FF D3(00)		
FF A4(00)	HEX$	FF D4(00)		
FF A5(80)	INSTR	FF D5(00)		
FF A6(80)	INT	FF D6(00)		
FF A7(80)	INP	FF D7(00)		
FF A8(00)	INKEY$	FF D8(00)		
FF A9(80)	LPOS	FF D9(00)		
FF AA(80)	LOG	FF DA(00)		
FF AB(80)	LOC	FF DB(00)		
FF AC(80)	LEN	FF DC(00)		
FF AD(00)	LEFT$	FF DD(00)		
FF AE(80)	LOF	FF DE(00)		
FF AF(00)	MKI$	FF DF(00)		

表 2-4-2 中間言語：キーワード

2-4-2-3 中間言語テーブル

中間言語はキーワードの1文字目のアルファベットによって分類され，各データはキーワードの文字数-1，キーワード，中間言語，フラグから成り立っています．

キーワードは1文字目がグループ分けのために最初に書かれているので省略され，その代わりにキーワードの先頭に，キーワードの文字数-1を示す1バイトの数字が置かれます．

中間コードは1バイトのものはそのままで，2バイトのものは(FF+XX)，最上位ビットを0にして1バイトで表されています．

2-4-3 ラベルテーブル

ラベルテーブルは，シンボルテーブルセグメント内にあり，セグメント60Hの6AEH，6AFHで示されるアドレスから6B0H，6B1Hで示されるアドレスまでがラベルテーブルです．

ラベルテーブルは以下に示すような形式でアルファベット順にソートされて格納されてます．



図2-4-2 ラベルテーブルの格納形式

最初の1バイトはラベルの頭文字で，次の1バイトは(ラベル名の文字数-1)を示し，次の部分は頭文字を除いたラベルの後の部分が入っています．最後の2バイトは，そのラベルのついた行の先頭アドレスを示しています．

また，8086の場合，最初の1バイトは必ず偶数にする必要があり，このためラベルテーブルの総バイト数を合わせて，次の行が必ず偶数アドレスから始まるように1バイト余分につけることがあります．

なぜ偶数にするか，というと8086は奇数アドレスをアクセスするより，偶数アドレスをアクセスした方が速いからです．

2-4-4 変数テーブル

2-4-4-1 単純変数テーブル

単純変数テーブルはラベルテーブルの後に作られ，シンボルテーブルセグメントのオフセット0002，0003Hから0004H，0005Hで示されるアドレスまでです．

プログラム中で使われる変数は，変数の型に応じて使われる順番に登録されます．

ここでもPC-8001やPC-8801シリーズとは違って，それぞれリンクポインタを持ち，リスト形式で頭文字のアルファベット順に並んでいます．

各変数の型による形式を以下に示します．

整数型	リンクポインタ	2	変数名	データ
単精度	リンクポインタ	4	変数名	データ
倍精度	リンクポインタ	8	変数名	データ
文字型	リンクポインタ	3	変数名	データ

図2-4-3　変数の格納形式

データの長さは整数型が2バイト，単精度型が4バイト，倍精度型が8バイト，文字型が4バイトで，文字型のデータはストリングディスクリプタです.

変数名は(変数名の長さ-1)と頭文字を取った変数名から成り，後には頭文字を取った変数名が続きます．ただし，1文字変数の時，変数名は0になります.

・ストリングディスクリプタ

最初の1バイトが文字列の長さを表し，次のバイトがリロケーションコードで，後に続く2バイトが文字列格納のオフセットアドレスを示します.

・リンクポインタ

データと変数の対応は，例えば頭文字がAである最初の変数の位置はシンボルテーブルセグメント(セグメント60Hの1410H, 1411H)のオフセット0〜FFHの256バイトのワークエリアの3CH, 3CDHに格納されています.頭文字がAである2番目の変数は，1番目の変数のリンクポインタによって指定された位置にあります．これを図に表すと次のようになります.

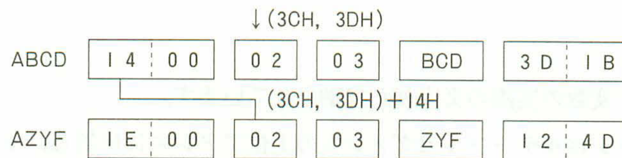


図2-4-4　データと変数の対応

というように，最初に出てくる変数のリンクポインタに次の同じ頭文字を持つ変数の位置が示されています.

2-4-4-2　配列変数テーブル

配列のデータは，セグメント60Hのセグメントポインタ(6A2H, 6A3H)の示すセグメントから格納されます.

配列名やその次数などは，前項の単純変数テーブルのすぐ後に作られます．このテーブルのアドレスは，セグメントポインタ(1410H，1411H)の示すシンボルテーブルセグメントのワークエリア(0004H，0005H)に格納され，頭文字のアルファベット順に並んでいますが，単純変数のよ

うなリスト構造ではなく，次の図に示すような構造となっています．( )の中の値はシンボルテーブルセグメントにあるワークエリアのポインタを表しています．

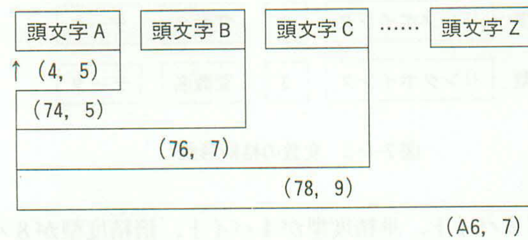


図2-4-5　配列変数テーブル

ポインタ(74 H，75 H)には頭文字がAである配列群の大きさが入っています．また，ポインタ(76 H，77 H)には頭文字がBである配列群の大きさに今までに出てきた配列群の大きさを加えたものが入っていおり，同様にZまで続きます．

配列群には，頭文字が等しい1つ，もしくは複数の配列の性質で構成され，配列の性質は次の図に示すような構造になっています．



図2-4-6　配列の構造

型や変数名は単純変数と同じで，次数は2バイト，配列数は(次数×2バイト)，配列要素の格納されるセグメントは2バイト，配列の大きさは2バイトとなっています．

2-4-5　文字エリア

文字エリアには文字変数の実際の文字列が格納されています．

文字列エリアは，シンボルテーブルセグメントの上位アドレスに位置し，その上限ポインタはテキストセグメント60 H内のワークエリア(6 B 2 H，6 B 3 H)，下限ポインタは(6 B 4 H，6 B 5 H)にあります．

このポインタの上限は固定されていて動かず，文字列が増やされると下限のポインタが更新されて下位アドレスの方向へ文字列が格納されていきます．

例えば，A＄という変数に〝ABCDE〟という文字列を代入し，B＄には〝@〟という文字を，さらにその後，A$の内容を〝1234〟という文字列に変更したとすると，文字エリアの内容は次の図のようになります．

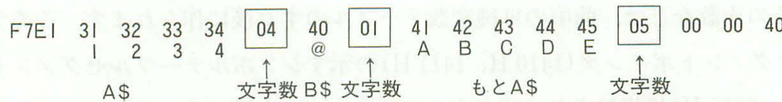


図2-4-7　文字エリア

この様に，最初に代入された〝ABCDE〟はメモリ上から消えずに残っており，ここで，A$=〝XYZ〟とやると，〝1234〟が追いやられて〝XYZ〟が新しい値となりますが，〝ABCDE〟も残っています．

ここでガベージコレクションを行う(FREコマンド)と，前の文字列は消去されて現在の内容である〝XYZ〟と〝@〟だけが文字列エリアの後ろから詰められます．

2-5 BASIC新コマンドの追加

2-5-1 未使用コマンドの使用

N 88-BASIC(86)では，キーワードとして中間コードが割り付けてあっても，実際には使用されていないコマンドがいくつかあります．新コマンドの追加の前に，これら未使用コマンドの定義方法について説明します．

これらの「あっても使用されないコマンド」は，例えばROM-BASIC使用時のDISK-BASICコマンドやGPIBインターフェースを使用していない時のGPIB関係のコマンドがあります．これらを普通に使用しようとしとても〝Feature not available〟になりますが，PC-8001やPC-8801の場合と同様にユーザーが定義して使用することが出来ます．GPIBのCMDコマンドを例として，未使用コマンドの定義方法について説明します．

2-5-1-1 フラグの飛び先のセット

N 88-BASIC(86)では，コマンド解析ルーチンで拡張コマンド使用時のフラグを見て，RAM上のアドレスをFAR CALLしています．1行実行ルーチンのフローチャートは次のようになっています．

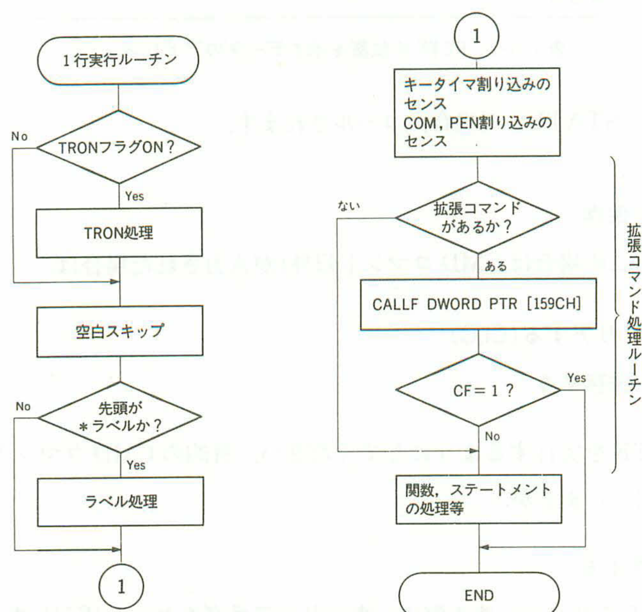


図2-5-1 フローチャート

このフローチャートで拡張コマンドがあるか否かの判断は，

セグメントベース　　60H

オフセットアドレス　1593H

の内容が，拡張コマンドを使用するならば0以外の値，使用しないならば0というようにセットされたフラグを見て行っています．

従って，CMDコマンドを使用するためには，まず次のようにセットする必要があります．

1593H	0以外の値をセット
159CH 159DH	CMD処理ルーチンのオフセットアドレスをセット
159EH 159FH	CMD処理ルーチンのセグメントアドレスをセット

表2-5-1　フラグの飛び先のセット

これはステートメントの処理であって，この他に関数処理ルーチンで拡張コマンドのフラグ(1593Hのフラグ)を見て，

CALL　DWORD　PTR　[15AH0]

としている部分があるため，さらに次のようにセットする必要があります．

15A0H 15A1H	RETF命令(コードCBH)のあるオフセットアドレス
15A2H 15A3H	RETF命令のあるセグメントベースアドレス

表2-5-2　RETFの位置を示すデータのアドレス

15A0HはIEEE，STATUSの場合，コールされます．

2-5-1-2　レジスタの保存

関係ないコマンド(この場合はCMDコマンド以外)が入力された場合は，

キャリーフラグをクリアする(CLC)

AL，SIレジスタを保存する

として，すぐにRETFを実行するようにしてください．目的のCMDコマンド(中間コードE8H)の場合は処理を実行しますが，

DSレジスタは保存する

処理が終ってRETFでリターンする際に，キャリーフラグをセット(STC)する

このように，それぞれの場合によってレジスタ等を保存する必要があります．

2-5-1-3 CMD ルーチンに入ってきたときの状態

10 CMD CLEAR KEY という行を実行したとき，SI レジスタの内容は，

6 E 8 H，6 E 9 H 実行中の行の先頭オフセットアドレス
6 EAH，6 EBH CMD

となっています．実際のテキストをダンプすると，

```
27 D 8 0 B 00 0 A 00 01 E 8 01 88 01 A 9 00 00 00 00 00 00
                        ↑
                        27 DD  E 8 H：CMD 88 H：CLEAR A 9 H：KEY
```

となっています．

次の〝：〟か文の END MARK の 00 のあるオフセットアドレスを (6 EAH，6 EBH) にセットして RETF する

以上を参考にして，ユーザーでコマンドを定義します．なお，N 88-DISKBASIC(86) の MON コマンドでは，簡易アセンブラが使用できますが，このアセンブラでは RETF や CALLF などのセグメント間コールをサポートしていません．そこで，RETF などの簡易アセンブラで入力できないニーモニックは，機械語コードの CBH を，S コマンドで直接入力することによってプログラムを記述します．

2-5-2 新コマンドの追加

キーワードとして登録されていない (ROM に焼かれていない) コマンドを作るにはどうすれば良いかをこの項で説明します．キーワード以外の文字列は，″で囲まれたものや REM 文以外は変数名としてテキストに置かれています．変数名としてテキストに置かれた文字列は次のような形になっています．

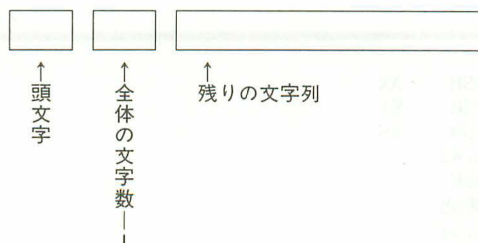


図 2-5-2 変数名としてテキストに置かれた文字

例えばBYEというコマンドを作ったとすると，このコマンドは，

```
B       02    YE
↑       ↑
AL      SI
```

という形になっており，先頭のBがALレジスタに入っている場合，そのあとに05 YEが連続して続いているならば処理ルーチンに入り，それ以外はキャリーフラグをクリアして，すぐにファーリターンするようにすればよいのです．レジスタの保存等については前項と同じです．

このBYEコマンドにソフトウェアリセット機能を付けた例を以下に示します．

プログラム2-5-1　BYE.BAS

```
10 '
20 '   "BYE.BAS"
30 '
40 '        *** Binal Program Loader for "BYE.BIN" ****
50 '
60 '                          update 1986.09.20 Programed by DARYL
70 '
80 '                               copyright (C) DMSC
90 '
100    DEF SEG=&H60 : POKE &H1593, 0              ' Expand  Command Flag Off
110 '
120    DEF SEG=&HA100 : BLOAD "BYE.BIN"           ' Load  Program
130 '
140   DEF SEG=&H60
150   ADR=&H159C :GOSUB *SET.PARA                ' Set Command Segement & Offset
160   ADR=&H15A0 :GOSUB *SET.PARA
170 '
180   POKE &H1593, 1                             ' Expand Command Flag On
190 '
200   PRINT "Command Created  BYE   For Reset"
210 '
220 NEW
230 END
240 '
250 *SET.PARA
260    POKE ADR,0: POKE ADR+1,0: POKE ADR+2,0: POKE ADR+3,&HA1
270   RETURN
```

プログラム2-5-2　BYE.ASM

```
0000 50               PUSH    AX
0001 56               PUSH    SI
0002 1E               PUSH    DS
0003 3C42      CMP    AL,42
0005 7527      JNE    002E
0007 AC               LODSB
0008 3C02      CMP    AL,02
000A 7522      JNE    002E
000C AC               LODSB
000F 751D      JNE    002E
```

```
0011 AC                 LODSB
0012 3C45      CMP      AL,45
0014 7518      JNE      002E
0016 AC                 LODSB
0017 3C00      CMP      AL,00
0019 7404      JE       001F
001B 3C0B      CMP      AL,0B
001D 72F7      JB       0016
001F 4E                 DEC     SI
0020 8936E806    MOV    [06E8],SI
0024 31F6      XOR      SI,SI
0026 BE80FD    MOV      SI,FD80
0029 8ECE      MOV      CS,SI
002B E953FD    JMP      FD80
002E F8                 CLC
002F 1F                 POP     DS
0030 5E                 POP     SI
0031 58        POP      AX
0032 CB                 RETF
```

注意：

BYEコマンド追加プログラムは，N 88-DISKBASIC(86) version 3.0上での動作のみ保証します．また，機械語プログラム〝BYE.BIN〟は，DISK-BASICを起動し，MONコマンドで機械語モニタモードにしてから，Aコマンドで入力して下さい．その場合のセグメントベースアドレスは，ユーザーメモリの量に応じて設定して下さい．

2-5-3 プログラムの説明

BASICのプログラムは，まずセグメント60 Hで拡張コマンドのフラグをOFFし，セグメントA 100 Hに機械語プログラムをロードします．ここはVRAMのアドレスですので，CLEAR文は必要ありません．その後，再びセグメント60 Hに戻り，BYEコマンドの処理ルーチンのオフセットアドレスを159 C，DHに，セグメントベースアドレスを159 E，FHにセットし，RETF命令(CBH)のあるオフセットとセグメントアドレスをセットします．そして，拡張コマンドのフラグ(1593 H)を立て，処理を終えます．

機械語プログラムは打ち込まれた文字がBかどうかを判断し，その後に2文字続くかどうかを判断して，Bに続く文字がY，Eならば次の処理をし，違うならばキャリーフラグをクリアして，AX，SI，DSレジスタを保存してRETFします．このRETF命令はモニタアセンブラにはない命令ですので，Sコマンド，またはEコマンドで，機械語コードCBとして直接打ち込んで下さい．

コマンドラインから打ち込まれた文字がBYEだと確認された場合，CSレジスタにFD 80 HをセットしてFD 80 HにJMPします．これで，ソフトウェアリセットが掛かるわけです．

N 88-BASIC(86)が起動した時に，この〝BYE.BAS〟が動作するように〝setinf.n 88〟を使ってセットしておけば，リセットしたい時にBYEと打ち込むだけでソフトウェアによるリセットがかかります．

ただし,このプログラムはDISK-BASICを起動した直後に実行するようにしてください.そうでないと予測されない動作をすることがあります.

このBYEというコマンドは変数として格納されているものを利用していますが,?BYEとしても内容は0です.また,BYE=123などとしようとしてもエラーになります.

高級言語と98V

第3章

3-1 MS-DOS版N 88-日本語BASIC(86) インタープリタ／コンパイラ

3-1-1 DISK-BASIC版との違い

この項では，MS-DOS版N 88-日本語BASIC(86) ver 3.0とDISK-BASIC版との機能の違いについて説明します．

MS-DOS版になって付加された機能には次のようなものがあります．

<標準でサポートされているもの>

1. チャイルド・プロセス制御機能
2. マウス操作機能
3. 階層化ディレクトリ構造への対応

<オプションでサポートされているもの>

4. GP-IB制御機能
5. MUSICジェネレータ制御機能
6. ネットワーク制御機能

また，削除された機能には次のようなものがあります．

1. ライトペン制御機能
2. カセットテープ制御機能
3. モード変更機能(MON，TERM)

以下で標準でサポートされている機能について解説します．

3-1-1-1 チャイルド・プロセス制御機能

チャイルド・プロセスを使う方法に関しては3-1-2で詳しく説明しています．そちらを参照してください．

3-1-1-2　マウス操作機能

MS-DOS版N88-日本語BASIC(86)ではマウス操作に関する命令／関数が標準でサポートされており，DISK-BASIC版のようにマウス・ドライバをロードしてCALLする，といった煩わしさが解消されています．次にマウス制御用の命令／関数について解説します．

MOUSE(命令)：マウスの種々の動作環境の規定

書　式
MOUSE 0
MOUSE 1〔,<Sx>〕〔,<Sy>〕〔,カーソル・スイッチ〕
MOUSE 2,<X 指示点>,<Y 指示点>,<XOR 文字列>
MOUSE 3,<方向>,<移動率>
MOUSE 4,<Sx 1>,<Sy 1>,<Sx 2>,<Sy 2>
MOUSE 5,<色>
MOUSE 6

解　説

MOUSE　0

マウス・カーソルの形状，移動比率，移動範囲を初期化します．MOUSE　0を実行しないで他のマウス命令／関数を実行するとエラーとなりますので注意が必要です．この命令ではマウス・カーソルは表示されません．

MOUSE　1

マウス・カーソルの位置をスクリーン座標上の(Sx，Sy)に設定します．省略した場合は現在の座標となります．カーソル・スイッチは1で表示，0で非表示となります．省略した場合は現在の状態が引き継がれます．なお，MOUSE　0の実行直後はマウス・カーソルは表示されませんが，座標位置は画面中央になっています．

MOUSE　2

マウス・カーソルの形状，指示点を設定します．マウス・カーソルの形状は16×32ドットの範囲内で任意に設定できます．また，指示点は16×32ドットの範囲内の任意の1ドットを選択することができます．<X 指示点>は0～15，<Y 指示点>は0～31の値で設定します．省略時の値は(0，0)です．マウス・カーソルの形状設定は図3-1-1のようにマウス・カーソルの1ドットのON／OFFを第4パラメータの<XOR 文字列>に対応させて行います．

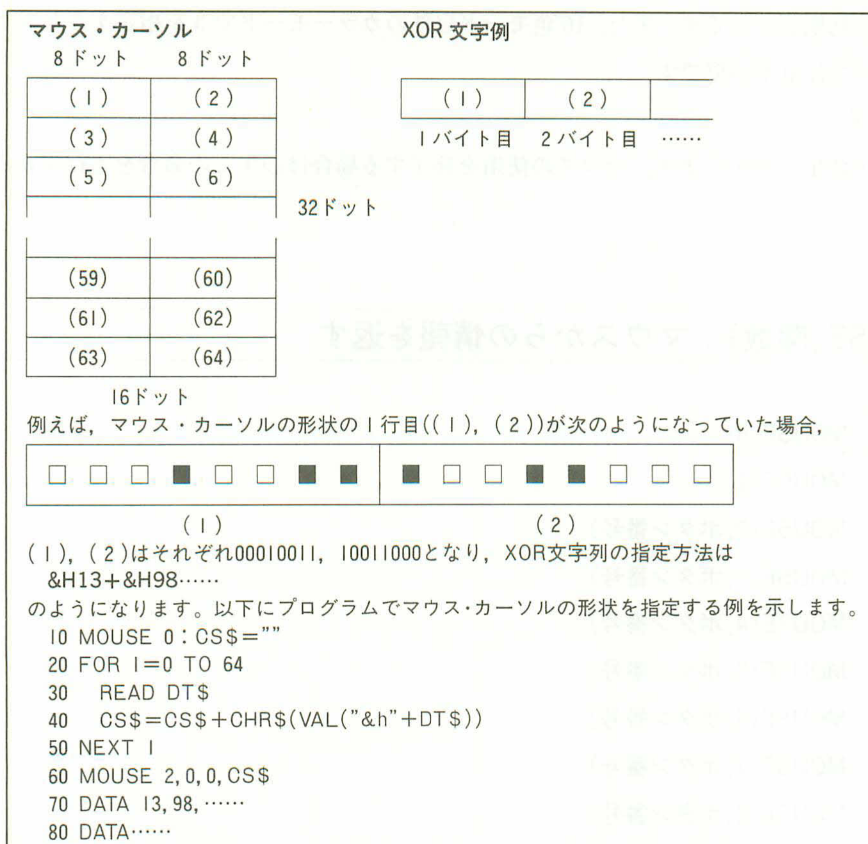


図3-1-1 マウス・カーソルの形状設定

MOUSE 3

マウス・カーソルの移動比率を設定します．マウス・カーソルを8ドット移動させるのに必要なマウスの移動距離を0～32767の整数で第3パラメータに設定します．初期値は8です．0を設定すると初期値に戻ります．第2パラメータの<方向>は0を設定するとX方向，1を設定するとY方向移動比率が変わります．

MOUSE 4

マウス・カーソルの移動範囲を設定します．(Sx 1，Sy 1)，(Sx 2，Sy 2)の2点を対角とする四角形がマウス・カーソルの移動範囲となります．マウスをこの範囲外に移動させても，マウス・カーソルは設定された範囲内に留まるようになります．

MOUSE 5

マウス・カーソルの色を指定します．指定する値とパレットの関係は次のようになっています．

0：パレット番号1
1：パレット番号2
2：パレット番号4
3：パレット番号8

<色>の初期値は2です．また，16色モード以外のカラーモードで3を指定するとエラーとなりますので注意が必要です．

MOUSE　6

マウスの使用を終了します．マウスの使用を終了する場合は必ずこの命令を実行する必要があります．

MOUSE(関数)：マウスからの情報を返す

書　式　MOUSE(0)

MOUSE(1)

MOUSE(2,ボタン番号)

MOUSE(3,ボタン番号)

MOUSE(4,ボタン番号)

MOUSE(5,ボタン番号)

MOUSE(6,ボタン番号)

MOUSE(7,ボタン番号)

MOUSE(8,ボタン番号)

MOUSE(9)

MOUSE(10)

解　説

MOUSE(0)

マウス・カーソルの現在のX座標を返します．

MOUSE(1)

マウス・カーソルの現在のY座標を返します．

MOUSE(2)

ボタン番号で指定されたボタンが押されている時は1を，押されていない時は0を返します．

MOUSE(3)

最後にこの関数が実行されてから現在までにボタン番号で指定されたボタンが押された回数を返します．

MOUSE(4)

ボタン番号で指定されたボタンが最後に押された時のX座標を返します．

MOUSE(5)

ボタン番号で指定されたボタンが最後に押された時のY座標を返します．

MOUSE(6)

最後にこの関数が実行されてから現在までにボタン番号で指定されたボタンが離された回数を返します.

MOUSE(7)

ボタン番号で指定されたボタンが最後に離された時のX座標を返します.

MOUSE(8)

ボタン番号で指定されたボタンが最後に離された時のY座標を返します.

MOUSE(9)

最後にこの関数が実行されてから現在までにマウス・カーソルが移動したX方向の移動距離を返します. 右が正の方向, 左が負の方向になります.

MOUSE(10)

最後にこの関数が実行されてから現在までにマウス・カーソルが移動したY方向の移動距離を返します. 下が正の方向, 上が負の方向になります.

MOUSE(n)　ON／OFF／STOP：マウスによる割り込みの許可／禁止／停止

書　式　MOUSE(事象番号) ON

MOUSE(事象番号) OFF

MOUSE(事象番号) STOP

解　説

事象番号の意味は次のようになっています.

1：マウスが移動した

2：左ボタンが押された

3：右ボタンが押された

4：左ボタンが離された

5：右ボタンが離された

MOUSE(n)　ON

事象番号で指定された割り込みを許可します. この命令が実行された後, 指定された事象が発生する度に割り込みが発生し, ON MOUSE GOSUB 文で指定されたサブルーチンに分岐します.

MOUSE(n)　OFF

事象番号で指定された割り込みを禁止します．この命令が実行された後，指定された事象が発生しても分岐はしません．

MOUSE(n)　STOP

事象番号で指定された割り込みを停止します．この命令が実行された後，指定された事象が発生しても分岐はしませんが，発生した事象は記憶されており，MOUSE(n)　ON で割り込みが許可された後，分岐します．

ON　MOUSE(n)　GOSUB：マウスによる割り込みが発生した時に分岐するサブルーチンを指定

書　式　ON MOUSE（事象番号）GOSUB 行番号

解　説

事象番号は〝MOUSE(n)　ON／OFF／STOP〞で示したものと同じです．行番号は指定された割り込みが発生したときに分岐するサブルーチンの開始行です．行番号にはラベルも使用できます．

3-1-1-3　階層化ディレクトリについて

DISK-BASIC 版 N 88-日本語 BASIC(86)ではディスク・ファイルの形式が BASIC 独自のものでしたが，MS-DOS 版では MS-DOS の形式になったため，階層ディレクトリをサポートしています．

次に階層ディレクトリィ支援のための命令(コマンド)について解説します．

CHDIR：ディスク・ファイルのカレント・ディレクトリの変更

書　式　CHDIR［ドライブ名］ディレクトリ名

解　説

ドライブ名で指定されたディスクのカレント・ディレクトリをディレクトリ名で指定されたディレクトリに変更します．ドライブ名が省略された場合はカレント・ドライブと見なされます．なお，カレント・ディレクトリのすぐ上のディレクトリに変更する場合，ディレクトリ名として〝..〞を使用することができます．

実　例

```
CHDIR　"B：¥MCIN¥PROG"
CHDIR　"B：.."
```

MKDIR：サブディレクトリの作成

書　式　MKDIR［ドライブ名］ディレクトリ名

解　説

ドライブ名で指定されたディスクにディレクトリ名で指定されたサブディレクトリを作成します．ドライブ名が省略された場合はカレント・ドライブと見なされます．

実　例

```
MKDIR　"YASUI"
RMDIR　"B：¥KUSUMI"
```

3-1-1-4　日本語処理機能について

●入力方法

MS-DOS 版 N 88-日本語 BASIC(86)では MS-DOS の日本語入力機能を使用していますので，日本語入力操作は MS-DOS と同じになります．

●内部コード

MS-DOS 版 N 88-日本語 BASIC(86)の日本語内部コードは MS-DOS と同じシフト JIS コードを使用しています．そのため，他の MS-DOS 上で動くアプリケーション等と日本語データの互換性があります．また，KI／KO コードを使う必要がなくなったために日本語処理用の関数の扱いが簡便になっていますが，日本語とグラフ文字の混在はできなくなっています．

日本語処理用の命令／関数には次のようなものがあります．

●命令

KINPUT，KPLOAD

●関数

AKCNV$，JIS$，KANCNV$，KEXT$，KINSTR，KLEN，KMID$，KNJ$，KTYPE

これらの命令／関数の扱いは DISK-BASIC 版 N 88-日本語 BASIC とほぼ同じです．

3-1-2　N88-日本語BASIC(86)でマシン語を使う方法

この項では，MS-DOS版N88-日本語BASIC(86)でマシン語を使う2通りの方法について説明します(サンプル・プログラムとして提供されているTEST.COM，SORT.COMの二つはMASMでアセンブルされたCOM形式のマシン語プログラムです)．

3-1-2-1　CHILD命令を使う方法

CHILD命令はBASICインタープリタのコマンド・レベル，またはプログラム・レベルから一時的にMS-DOSに制御を移し，指定されたプロセスを実行した後，BASICに制御を戻します．

● CHILD　"TEST.COM"，M

この方法はCOMMAND.COMを起動してからチャイルド・プロセス"TEST.COM"を作動させます．BASICプログラム中に，この方法(Mオプションを付ける)を使用する場合，チャイルド・プロセスはBASICのシンボル・テーブル以後にロードされて実行されますので，BASICプログラムの実行は中断されます．

● CLEAR,,,&H1000　：　CHILD　"TEST．COM"

この方法はチャイルド・プロセス領域を確保してからチャイルド・プロセス"TEST.COM"を作動させます．Mオプションが付いていないので，チャイルド・プロセスはCLEAR命令で確保された専用領域にロードされて実行されますので，BASICプログラム中で使用してもBASICプログラムの実行には差し支えありません．

次に簡単なマシン語プログラムをチャイルド・プロセスの例として上げておきます．

プログラム3-1-1　TEST.ASM

```
; test.asm
;
;        ＢＡＳＩＣとのリンクテストプログラム
;        update 1986.08.01 copyright (C) DMSC
;
;        このプログラムのソースリストを表示します
;        単体でも、作動する。ＭＡＣＲＯ設定

msdos    equu       0

com_model           macro
         code       segment
                    assume  cs:code,ds:code,es:code,ss:code
                    org     0100h
         prog:
                    endm

com_end             macro
         code       ends
                    end     prog
                    endm
;

dosdx               macro   func,data
                    local   data_adrs,l1
```

```
                mov     ax,func
                mov     dx,offset data_adrs
                int     21h
                jmp     l1
data_adrs:
                db      data
l1:
                endm
;
print           macro   data
                dosdx   0900h,<data>
                endm
;
open            macro   data
                dosdx   3d00h,<data>
                endm
;
;
;               プログラム
;

        com_model                ; com model start up macro

        public  non_error1,read_loop,non_error2,_exit,buffer

        print <0dh,0ah,'SAMPLE PROGRAM. DISPLAY TEST.ASM.',0DH,0AH,'$'>

        open <'TEST.ASM',0>

        jnc     non_error1
        jmp     _exit
non_error1:
        mov     bx,ax

;       read file

read_loop:
        mov     dx,offset buffer
        mov     cx,512
        mov     ah,3fh
        int     21h

        jnc     non_error2
        jmp     _exit
non_error2:
        and     ax,ax
        jz      _exit

        mov     cx,ax            ; print out
        mov     ah,40h
        push    bx
        mov     bx,1
        int     21h
        pop     bx

        jmp     read_loop
_exit:

        ret

buffer: db      513 dup (?)

        com_end                  ; com_model close macro
```

アセンブルは次のようにして行います．

```
MASM  TEST;
LINK  TEST;
EXE2BIN  TEST.EXE  TEST.COM
```

CHILD命令を使う場合，チャイルド・プロセスはCOMモデル，EXEモデルのいずれの形式のプログラムでも使用可能です．

3-1-2-2 CALL命令を使う方法

この方法はBASICプログラム中にマシン語プログラムをサブルーチンとしてロード，実行します．このほか，USR関数を使用してマシン語プログラムを関数としてロード，実行する方法もありますが，引数を1つしか持てないなどの欠点があるため，CALL命令を使う方法が一般的といえるでしょう．

CALL命令でマシン語をコールする手順を次に示します．

```
1.  CLEAR &H1000              ……マシン語プログラム領域の確保
2.  DEF SEG = SEGPTR(2)       ……マシン語領域のセグメント指定
3.  BLOAD "SORT.COM",&H100    ……マシン語プログラムのロード
4.  SORT =&H100               ……実行開始番地の設定
5.  CALL SORT                 ……マシン語プログラムの実行
```

次に各手順について解説します．

1. CLEAR文でマシン語領域を確保するには，第一パラメータに確保するバイト数を16で割った値を指定します．余りが出る場合は，1余分に指定します．グラフィック画面などをマシン語領域に使う場合には，確保は必要ありません．
2. CLEAR文で確保したマシン語領域は，SEGPTR関数を使用してセグメント指定をする必要があります．これはBLOAD，BSAVEで制御できるアドレスがオフセット・アドレスだけであるためです．SEGPTR関数の機能はインタープリタとコンパイラで多少の違いがあるので注意が必要です．以下にその違いを示します．

<インタープリタ>

SEGPTR(0)：メモリの上限
SEGPTR(1)：チャイルド・プロセス領域の先頭
SEGPTR(2)：マシン語プログラム領域の先頭
SEGPTR(3)：配列データ領域の先頭
SEGPTR(4)：文字列演算用作業域の先頭

SEGPTR(5)：シンボル・テーブル／文字列領域の先頭
SEGPTR(6)：プログラム領域の先頭
SEGPTR(7)：システム制御情報領域の先頭
SEGPTR(8)：エラー
<コンパイラ>
SEGPTR(0)：メモリの上限
SEGPTR(1)：チャイルド・プロセス領域の先頭
SEGPTR(2)：マシン語プログラム領域の先頭
SEGPTR(3)：配列データ領域の先頭
SEGPTR(4)：エラー
SEGPTR(5)：エラー
SEGPTR(6)：エラー
SEGPTR(7)：システム制御情報領域の先頭
SEGPTR(8)：定数領域の先頭

3. マシン語プログラムのロード．ロード・アドレスをパラメータ指定します．指定しない場合，0000 H からロードされます．
4. SORT＝&H 100 で実行開始番地の指定を行います．
5. CALL＝SORT でマシン語プログラム〝SORT.BAS〞を実行します．

CALL 命令を使う場合，マシン語サブルーチンは COM モデルのプログラムでなければ正常に動作しないので注意が必要です．

次に簡単なマシン語プログラムをサブルーチンの例として上げておきます．なお，このプログラム中で include されるライブラリー・ファイル，〝commodel.lib〞は前出のサンプル・プログラム，〝TEST.ASM〞のマクロ設定の部分と同じものです．

プログラム 3-1-2 SORT.ASM

```
; sort.asm
;
; this is compiled program by HI-TECH C Compiler
;       c -s -o sort.c
;                       update 1986.08.01 copyright (C) DMSC
;
; このプログラムはC言語で書かれたプログラムをコンパイルして出力された
; アセンブルリストに手を加えて、作られたものです。

include commodel.lib

        com_model

_tsort:
        push    si
```

```
        push    di
        push    bp
        mov     ax,ds
        push    cs
        pop     ds

        mov     bp,sp
        sub     sp,6                    ; int   i,j,k;
        push    ax
        mov     word ptr -2[bp],0       ; for (i=0; i<num; i++)
        jmp     l5
l2:
        mov     word ptr -4[bp],0       ; for (j=0; j<num-1; i++)
        jmp     l9
l6:
        mov     bx,-4[bp]               ; if (*(adrs+j)>*(adrs+j+1)) {
        sal     bx,1
        add     bx,cs:[0080h]
        mov     ax,[bx]
        mov     bx,-4[bp]
        sal     bx,1
        add     bx,cs:[0080h]
        cmp     ax,2[bx]
        jle     l8
        mov     bx,-4[bp]               ; k=*(adrs+j);
        sal     bx,1
        add     bx,cs:[0080h]
        mov     ax,[bx]
        mov     -6[bp],ax
        mov     bx,-4[bp]               ; *(adrs+j)=*(adrs+j+1);
        sal     bx,1
        add     bx,cs:[0080h]
        mov     ax,2[bx]
        mov     bx,-4[bp]
        sal     bx,1
        add     bx,cs:[0080h]
        mov     [bx],ax
        mov     ax,-6[bp]               ; *(adrs+j+1)=k;
        mov     bx,-4[bp]
        sal     bx,1
        add     bx,cs:[0080h]
        mov     2[bx],ax
l8:
        inc     word ptr -4[bp]
l9:
        mov     ax,cs:[0082h]
        dec     ax
        cmp     ax,-4[bp]
        jnle    l6
        inc     word ptr -2[bp]
l5:
        mov     ax,-2[bp]
        cmp     ax,cs:[0082h]
        jl      l2

        pop     ds
        mov     sp,bp

        pop     bp
        pop     di
        pop     si

        iret

        com_end
```

アセンブルは次のようにして行います.

MASM SORT ;

LINK SORT ;

EXE 2 BIN SORT.EXE SORT.COM

上記のマシン語プログラムをリンクして, 作動させるBASICプログラムを次に示します. このプログラムではPOKE文を使用してマシン語サブルーチンに直接, 引き数のデータを書き込んでいます.

プログラム 3-1-3 SORT.BAS

```
100 ' sort.bas
103 '  sort.com test program
105 ' update 1986.08.01 copyright (C) DMSC
110 '
120 CLEAR &H1000                         ' マシン語領域の確保
130 DEFINT A-Z                           ' このプログラムは整数型ｓｏｒｔ
140 DEF SEG=SEGPTR(2)                    ' 確保したマシン語領域をセグメント指定
150 BLOAD "sort.com",&H100               ' ｓｏｒｔプログラムをロード
160 '                                      ロードアドレスを指定しないと
165 '                                      ００００ｈより、ロードされます
170 NUM=10                               ' ｓｏｒｔする。数値を１０個とする
180 BASE.ADRS=&H4000                     ' データ領域を＆ｈ４０００よりとする
190 '
200 PRINT "source."
210 '
220 FOR I=0 TO NUM-1                     ' データをメモリに格納する
230   S=RND*1000
240   POKE BASE.ADRS+I*2  ,S MOD 256     ' ８０系では、１６ビットデータは
250   POKE BASE.ADRS+I*2+1,S ¥   256     ' 下位、上位の順に格納される
260   PRINT S
270 NEXT
280 '
290 PRINT
300 '
310 POKE &H80,BASE.ADRS MOD 256          ' データの格納アドレスを８０ｈに格納
320 POKE &H81,BASE.ADRS ¥   256
330 '
340 POKE &H82,NUM MOD 256                ' データ数を８２ｈに格納する
350 POKE &H83,NUM ¥   256
360 '
370 SORT=&H100                           ' ｓｏｒｔプログラムは、１００ｈに
380 '                                      ロードされている
390 CALL SORT                            ' ｓｏｒｔ実行
400 '
410 PRINT "Result."                      ' 結果の表示
420 FOR I=0 TO NUM-1
430   PRINT PEEK(&H4000+I*2)+PEEK(&H4000+I*2+1)*256
440 NEXT
450 '
460 PRINT
470 '
480 END
```

MS-DOS版BASICではDISK-BASIC版と違い，BSAVEされたマシン語プログラムの先頭にロード・アドレス，ファイル・サイズが入らないので注意が必要です．

3-2　Lattice C ver 3.0

3-2-1　Lattice C ver 3.0の特徴

C言語は高級言語でありながら，かなりアセンブラに近い記述が可能なため，OS上で作動するユーティリティ・プログラムやOSそれ自体等，システマティックなプログラミングに向いています．しかし，もっとも低レベルのハードに依存した部分まではサポートしきれていないため，完全に機械語なしでプログラミングすることはできません．むしろ，C言語は機械語とのリンクの機会が多い言語であるといえるでしょう．

Lattice C ver 3.0は1986年3月にリリースされました．このバージョンアップによって，以下に示すような点が，ANSI標準化案に準じて改良されました．

・void型，enum型の導入
・構造体／共有体の扱い
・プロトタイプ宣言

また，MS-DOS ver 3.1に対応して，ネットワーク関連の関数がライブラリに加えられ，UNIXライクな関数も大幅に追加されています．

3-2-2　Lattice Cからの機械語コール

この項では，Lattice Cでマシン語を使う方法について説明します．

3-2-2-1　8086の割り込みを使う方法

割り込みを使う方法は，マウスドライバ等で使われています．割り込みNo.を指定して，メインのプログラムとは独立させて作動させる場合に使用します．この場合の利点は，一度プログラムをロードし，メモリに常駐させておけば複数のプログラムから呼ぶことができ，プログラムをロードするにもその分，時間が短くて済みます．

しかし，プログラムを常駐させておくのは手間のかかることであり，また，その分メモリ・エリアが少なくなるという欠点もあります．

3-2-2-2　機械語プログラムとリンクする方法

MASM等で書いた機械語プログラムを直接リンクする方法は，そのプログラムだけに必要な機械語サブルーチンを使用する場合に使います．

各レジスタを保存し，リターン値をAXレジスタにいれてretします．前述の方法に比べると，

非常に簡単に実現できます.

次にアセンブラで機械語サブルーチン(関数)を記述する具体的な方法を説明します.

●マクロ定義ファイル

Lattice Cではセグメント定義の簡略化のために,マクロ定義ファイルが供給されています.S,P,D,L各メモリモデル,及びCOMモデル用に別のファイルになっており,順に〝SM 8086.MAC〃,〝PM 8086.MAC〃,〝DM 8086.MAC〃,〝LM 8086.MAC〃,〝CM 8086.MAC〃というファイル名になっています.これらのファイルは条件判定のスイッチ以外は同じ内容で,表3-2-1に示すマクロが定義されています.

マクロ名	定義内容
DSEG	データ・セグメントの開始
ENDDS	データ・セグメントの終了
PSEG	コード・セグメントの開始
ENDPS	コード・セグメントの終了

表3-2-1　マクロ定義ファイルの内容

これらのマクロ定義ファイルをプログラムの先頭にinclude疑似命令で取り込んでおけば,どのメモリモデルでも同じマクロ名で参照することができます.なお,include疑似命令で取り込むファイルのファイル名を〝DOS.MAC〃としておき,メモリモデルに応じて〝SM 8086.MAC〃・・・・の方を〝DOS.MAC〃と変更する方が望ましいでしょう.

●プロシージャの定義

機械語サブルーチン(関数)の部分は,public疑似命令でプロシージャ名(関数名)を他のモジュール(C言語のプログラム)から呼び出せるようにします.なお,プロシージャの型は各メモリモデルによって異なります.

●イニシャライズ

関数に制御が移った場合,引数はスタック領域に置かれ,BPレジスタを参照して引数領域をアクセスします.BPレジスタは常にローカル変数領域のアドレスを保持しているので,これをスタックに保存し,SPレジスタの値をBPレジスタにコピーすることで作業領域をデータ・セグメント内の固定領域に設定します.

●レジスタの保存

C言語ではBPレジスタ,DSレジスタはそれぞれローカル変数領域,静的変数領域のアドレスを保持しており,関数内でこれらのレジスタの値を変更すると関数からリターンした後,変数領域,静的変数領域を正しくアクセスすることができなくなってしまいます.これらのレジスタを使用する場合は,スタックに保存しておく必要があります.また,SSレジスタやSモデル,Pモデルの場合にはESレジスタも保存する必要があります.

●機械語サブルーチンからのリターン

関数の処理が終ったら，保存していたレジスタを戻してメインルーチンに戻ります．

次に機械語のサンプル・プログラムを上げておきます．

プログラム 3-2-1　instr.asm

```
; instr.asm
;
;       文字列検索ユーティリィティ  INSTR
;       update 1986.08.09 copyright (C) DMSC
;
;       int     instr(strings1,strings2);
;       char    *strings1,*strings2;
;

        include DOS.MAC                 ; マシン語サブルーチン用マクロライブラリ

        if      LPROG                   ; ラージモデル、スモールモデルの違いを
offs    =       6                       ; 吸収する。
        else
offs    =       4
        endif

        pseg                            ; セグメント定義マクロ

        public  instr                   ; Cのプログラムから、呼び出せるように
                                        ; public 指定を行う。

        if      LPROG                   ; このサブルーチン（Cのプログラムから見れば
instr   proc    far                     ; 関数であるが）を変数 LPROG（ラージモデルで
        else                            ; 真となっている）により、ラージモデル、スモ
instr   proc    near                    ; ールモデルの両方で DOS.MAC へのリネームによ
        endif                           ; りアセンブルしなおすだけで利用出来るように
                                        ; 設定しておく。

        push    bp                      ; C言語の標準的なイニシャライズ
        mov     bp,sp
        push    si                      ; 使用するレジスタの保存
        push    di
        push    ds
        push    es

        if      LDATA
        mov     ax,offs[bp]             ; get arg1'seg
        mov     ds,ax
        mov     si,(offs+2)[bp]         ; get arg1'offset
        mov     ax,(offs+4)[bp]         ; get arg2'seg
        mov     es,ax
        mov     di,(offs+6)[bp]         ; get arg2'offset
        else
        mov     ax,ds
        mov     es,ax
        mov     si,offs[bp]             ; get arg1'offset
        mov     di,(offs+2)[bp]         ; get arg2'offset
        endif

        mov     bx,0
        mov     cx,di
loop:
        mov     ah,ds:[si]
        mov     al,es:[di]
        or      ah,ah                   ; 調べる文字列が終了した。
```

```
        jz      loopout1
        or      al,al               ; 検索パターンが終了した。
        jz      loopout2
        inc     si                  ; 文字を1文字ずつチェックする。
        inc     di                  ; パターンが一致しない場合、その前まで
        cmp     ah,al               ; パターンが一致していたならその先頭の
        jz      lavl                ; アドレスより検索をやり直す。そうで無
        mov     di,cx               ; い場合は次の文字へ進む。
        or      bx,bx
        jz      loop
        mov     si,bx
        mov     bx,0
        jmp     loop
lavl:                               ; パターンが一致した
        or      bx,bx               ; パターン一致の先頭をｂｘに保存する
        jnz     loop                ; ｂｘが０で無い場合２文字め以降だから
        mov     bx,si               ; そのままループする。
        jmp     loop
loopout1:
        or      al,al               ; パターンが一致しないで終了した場合
        jz      loopout2; リターン値として０を返す。
        mov     bx,0
loopout2:
        mov     ax,bx               ; パターンが一致した場合、そのアドレスを
                                    ; リターン値として返す。
        pop     es                  ; 保存していたレジスタをもとに戻す。
        pop     ds
        pop     di
        pop     si

        mov     sp,bp
        pop     bp
        ret

instr endp
        endps                       ; コード終了マクロ

        END
```

instr.asm のアセンブル手順は，以下のアセンブル作業時の画面のハードコピーを参照してください。

```
A>copy a:SM8086.MAC b:DOS.MAC          ←注1
      1 個のファイルをコピーしました

A>masm instr.asm;

The Microsoft MACRO Assembler , Version 1.27
 (C) Copyright Microsoft Corp 1981,1984


Warning Severe
ErrorsErrors
0       0

A>
```

注1：この場合は，Ａドライブに Lattice C のディスクを，Ｂドライブにプログラムディスクをセットして DOS.MAC ファイルを作成しています．コピーが終了したら，Ａドライブにプログラムディスクをセットしてください．尚，プログラムディスクには，instr.asm, masm.exe が入っていなければなりません．

画面 3-2-1　instr.txt

このプログラムは，BASICでよく使われるinstrをアセンブラで書いたものです．引数1に被検索ポインタを，引数2に検索文字列ポインタを設定して，callします．

検索文字列が見つかった場合は，そのオフセットを返します．見つからなかった場合には，0を返します．ただし，int型を返しますので，データ型が64k以上のメモリモデルを使用するときには，0かそうでないかで判断します．

次に，この機械語プログラムをコールするC言語のサンプル・プログラムを上げます．

プログラム3-2-2 test.c

```
/* test.c
 *
 *    関数ｉｎｓｔｒのｔｅｓｔ用プログラム （文字列の検索を行います。）
 *              update 1986.08.09 copyright (C) DMSC
 */

main()
{
        char    in[80],patt[80];

        printf("¥nInstr check program¥ninput source line:");
        scanf ("%s",in);
        printf("input pattern line:");
        scanf ("%s",patt);
        printf("%s in %s .... ?¥n",patt,in);
        if    (instr(in,patt))  printf("found.¥n");
        else                    printf("not found.¥n");
        exit(0);
}
```

test.cのコンパイル手順は，以下のコンパイル作業時の画面のハードコピーを参照して下さい．

```
A>lc test
Lattice MS-DOS C Compiler, Version 3.00
Copyright (C) 1985 Lattice, inc. All rights reserved.

Compiling TEST.C
TEST.c 20 Warnig 85: function return value mismatch
Module size P=0076 D=0067 U=0000

Total files: 1, Successful compilation: 1


A>link cs test instr,test,nul,lcs

Microsoft 8086 Object Linker
Version 3.00 (C) Copylight Microsoft Corp 1983,1984,1985
```

画面3-2-2 testc.txt

最初に被検索文字列を入力し，次に検索文字列を入力すると結果が表示されます．

3-3 TURBO PASCAL version 3.0

3-3-1 TURBO PASCAL の特徴

PASCAL というプログラミング言語は，1971年にスイスのニクラス・ワース博士によって設計，開発されたものです．当時，プログラミング言語として最も整っていたALGOL を元にAL-GOLW を設計し，さらにそれに手を加えた言語をPASCAL として発表しました．PASCAL が発表されたのは，構造化プログラミングという考え方が提唱された時とほとんど同時期であり，PASCAL の言語仕様の中にも，構造化の考え方の影響が多く見られます．

TURBO PASCAL は，このワース博士の設計した標準PASCAL の機能をほとんど含みながら，さらにパーソナル・コンピュータでの仕様を前提としてメモリーやI/O などのダイレクト・アクセスや文字列操作などの機能を拡張したPASCAL コンパイラです．また，コンパイルをメモリ上で行うため，コンパイル・スピードが極端に速く，しかも機能の高いスクリーン・エディタが組み込まれているので，修正・テストランを繰り返すプログラム開発作業がとてもスムースに行えます．そのため，高度なプログラムを開発するプロフェッショナル・プログラマの中にも，TURBO PASCAL を愛用する人が多くなってきています．

3-3-2 TURBO PASCAL による機械語サブルーチンの利用

TURBO PASCAL はパーソナルコンピュータ用の開発言語として設計されていますので，わざわざアセンブリ言語で記述しなくとも，低レベルの処理(個々のハードウェアに依存した処理)を実行することができます．

3-3-2-1 MS-DOS システムコールの利用

N 88-BASIC(MS-DOS 版)等でMS-DOS のシステムコールを使用することはできませんが，TURBO PASCAL であれば，短いコマンドで実行することができます．MS-DOS のシステムコールには，便利なファンクションが数多くあります．どのようなファンクションがあるかは，3-4-2 のMS-DOS システムコールの利用法を参照してください．

プログラム 3-3-1 CALE.PAS

```
(* cale.pas ******************************************************************)
(*      日付表示プログラム（ＭＳ－ＤＯＳファンクションコールのテスト）      *)
(*              update 1986.09.17  copyright (C) DMSC                        *)
(*****************************************************************************)

program cale;

type register=record
       ax,bx,cx,dx,bp,si,di,ds,es,flags:integer;
end;
```

```
var reg:register;
    y,m,d,youbi,counter:integer;

begin
      reg.ax:=$2a00;
      msdos(reg);
      y:=trunc(reg.cx);
      m:=trunc(reg.dx/256);
      d:=reg.dx mod 256;
      youbi:=trunc(reg.ax mod 256);

      write(' ',y,' 年  ',m,' 月  ',d,' 日 ');

              case youbi of
                      0: writeln(' 日曜日');
                      1: writeln(' 月曜日');
                      2: writeln(' 火曜日');
                      3: writeln(' 水曜日');
                      4: writeln(' 木曜日');
                      5: writeln(' 金曜日');
                      6: writeln(' 土曜日');
              end;
end.
```

システムコールは，アセンブリ言語で使用するレジスタを通して引数の受け渡しをします．そのため，あらかじめレジスタを整数型変数として宣言しなくてはなりません．

MS-DOSのシステムコールを呼び出しているのは，

msdos(reg)；

の部分です．ここでは，regという変数に日付を取得するファンクションナンバーである2C(16進数)を代入して引数としています．

システムコールによって，取得された結果を加工して画面に出力しています．

```
A>cale
 1986 年  9 月  17 日  水曜日

A>
```

画面3-3-1　CALE.TXT

3-3-2-2　メモリのダイレクトアクセス

TURBO PASCALでは特定のメモリ番地を参照するために，絶対変数が用意されています．

・1バイト単位のアクセス：mem[セグメントベースアドレス,オフセットアドレス]：=データ

・1ワード単位のアクセス：memw[セグメントベースアドレス,オフセットアドレス]：=データ

変数を mem,memw の前において，integer：= mem[$xxxx,$oooo]；とすることによって，メモリの内容を読み出すこともできます．

アドレスの指定は，cseg,dseg などの絶対番地関数を使用することもできます．従って，簡単なメモリーアクセスの場合は，わざわざアセンブリ言語によるプログラムを作成する必要はありません．

3-3-2-3 I／O ポートアクセス

アセンブリ言語で記述されることが多い処理として，I／O ポートアクセスもあげることができます．TURBO PASCAL では，この I／O ポートアクセスのために，port,portw という定義済み配列が用意されています．

port,portw は，mem,memw と同じ関係をもち，I／O ポートに出力するデータがバイト単位であるかワード単位であるかの違いだけです．

3-3-2-4 外部機械語プログラムの利用

以上の方法を使ってもプログラムを記述できない場合，アセンブリ言語で書かれたサブプログラムを作成することになります．もしくは，以前にアセンブリ言語などで記述されたプログラムを使用する場合は，TURBO PASCAL で再度記述するのは二度手間になるので，この方法を使います．

ここでは，グラフィック画面の消去を特殊な方法で行う機械語プログラムをあらかじめ作成しておき，TURBO PASCAL によってプログラムを呼び出すという処理を行います．

まず，機械語プログラムから説明します．

TURBO PASCAL で呼び出される外部機械語プログラムは，BP, CS, DS, SS レジスタの内容を保証しなくてはなりません．TURBO PASCAL との変数の受渡しは AX レジスタによって行われますので，引数を必要とするプログラムを作成する場合は，そのことを念頭に入れて設計してください．

プログラム 3-3-2 GCLS.ASM

```
; gcls.asm
;                     特殊ＧＶＲＡＭクリアプログラム
;                 update 1986.09.21 copyright (C) DMSC
;
code   segment
       assume  cs:code,ds:code,es:code,ss:code
;
       org 100h
;
start:
;
main   proc    near
       push    bp
       mov     bp,sp
```

```
        mov     ah,40h                  ;グラフィック画面の表示ON
        int     18h                     ;MS-DOSシステムコール
        call    paint                   ;画面を白に塗りつぶします
        call    cls                     ;特殊GRAMクリアサブルーチンをコール
        mov     ah,41h                  ;グラフィック画面表示OFF
        int     18h                     ;MS-DOSシステムコール
        mov     sp,bp
        pop     bp
        ret                             ;呼び出しルーチンに戻る
main    endp

;

        paint   proc
                push    di
                mov     ax,0a800h       ;青のプレーン
                mov     es,ax
                mov     ax,0ffffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
                mov     ax,0b000h       ;赤のプレーン
                mov     es,ax
                mov     ax,0ffffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
                mov     ax,0B800h       ;緑のプレーン
                mov     es,ax
                mov     ax,0ffffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
                pop     di
                ret
        paint   endp
;
        cls     proc
                push    es
                push    di
;
                mov     ax,0a800h       ;青のプレーン1
                mov     es,ax
                mov     ax,00fffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
                mov     ax,0b000h       ;赤のプレーン1
                mov     es,ax
                mov     ax,00fffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
                mov     ax,0b800h       ;緑のプレーン1
                mov     es,ax
                mov     ax,00fffh
                mov     cx,03fffh
                mov     di,0
                cld
                rep     stosw
```

```
;
        mov     ax,0a800h       ;青のプレーン2
        mov     es,ax
        mov     ax,000ffh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b000h       ;赤のプレーン2
        mov     es,ax
        mov     ax,000ffh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b800h       ;緑のプレーン2
        mov     es,ax
        mov     ax,000ffh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
;
        mov     ax,0a800h       ;青のプレーン3
        mov     es,ax
        mov     ax,0000fh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b000h       ;赤のプレーン3
        mov     es,ax
        mov     ax,0000fh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b800h       ;緑のプレーン3
        mov     es,ax
        mov     ax,0000fh
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
;
        mov     ax,0a800h       ;青のプレーン4
        mov     es,ax
        mov     ax,0
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b000h       ;赤のプレーン4
        mov     es,ax
        mov     ax,0
        mov     cx,03fffh
        mov     di,0
        cld
        rep     stosw
        mov     ax,0b800h       ;緑のプレーン4
        mov     es,ax
        mov     ax,0
        mov     cx,03fffh
        mov     di,0
```

```
                cld
                rep     stosw
;
                pop     di
                pop     es
                ret
        cls     endp
;
;
code    ends
        end     start
```

MS-DOSのマクロアセンブラのプログラム記述方法に関しては，専門の参考書籍を参照してください．このプログラムは引数を使用していませんので，単体でも作動します．

"GCLS.ASM"のコンパイル手順は，以下のコンパイル作業時の画面のハードコピーを参照してください．

```
A>masm gcls;

The Microsoft MACRO Assembler , Version 1.27
 (C) Copyright Microsoft Corp 1981,1984


Warning Severe
ErrorsErrors
0      0

A>link gcls;

Microsoft 8086 Object Linker
Version 2.44  (C) Copyright Microsoft Corp 1983

Warning: No STACK segment

A>exe2bin gcls.exe gcls.com

A>
```

画面 3-3-2　GCLS.TXT

続いてTURBO PASCALによる呼び出しプログラムを作成します．

プログラム 3-3-3　GRCLS.PAS

```
(* grcls.pas ***************************************************************)
(*                          特殊ＧＶＲＡＭクリア                              *)
(*              update 1986.09.17 copyright (C) DMSC                        *)
(***************************************************************************)

program grcls;
      var     i:integer;
      function gcls:integer;external'gcls';

      begin
              writeln('  特殊なグラフィック画面クリアです');
              i:=gcls;
end.
```

外部プログラムはプログラムの冒頭で定義しています.

```
function gcls : integer ; external'gcls' ;
```

この場合, gcls.com という外部プログラムが呼び出されます. 外部プログラムの拡張子は, MS-DOS の場合 ".COM" が初期値(システムによって設定されている値)ですが, その他の拡張子を指定することもできます. また, 呼び出す外部プログラムのパスが指定されていない場合, カレントディレクトリから外部プログラムを読み込みます. 外部プログラムが, そのディレクトリに存在しない場合, コンパイル時にエラーとなります. また, この部分を,

```
procedure gcls : integer ; external'gcls'
```

と, することもできます. 実際に外部プログラムを呼び出しているのは,

```
i : = gcls ;
```

の部分です. 整数型変数として型定義された i は, このプログラムの場合ダミーの引数として使われています. 引数を使う場合は, この変数によって引数の受け渡しを行います. コンパイルの方法は cale.pas と同じく, 特殊な方法は使用していません.

3-3-2-5　インライン機械語

TURBO PASCAL のプログラム中に, 短い機械語なら直接書き込んでしまうこともできます. この手法を, インライン機械語と呼びます.

```
begin
      inline($xx／xx・・・・・) ;
end ;
```

というように機械語コードの頭に$を付けて, 1バイトごとに／で区切りながら書き込みます. begin から end ; までの間をプロシージャとして定義してしまえば, 通常のプログラム構造として使用することができます.

3-3-2-6　絶対番地関数

メモリのダイレクトアクセスの項(3-3-2-2)でも少し触れましたが, TURBO PASCAL には, 機械語プログラムの利用をスムースに行うために, 絶対番地関数を用意しています. その種類と機能は, 以下に示す通りです.

seg　……　dumy という変数名を付けられたポインタ型変数のセグメントアドレス
ofs　……　dumy という変数名を付けられたポインタ型変数のオフセットアドレス

cseg　……　cs(コードセグメント・レジスタ)の内容

dseg　……　ds(データセグメント・レジスタ)の内容

sseg　……　ss(スタックセグメント・レジスタ)の内容

●注意

この節では，MS-DOS ver 2.11 上で TURBOPASCAL version 3.0 を使用してプログラムを作成しております．プログラム・リストを打ち込んで実行させるためには，以上の2点のソフトウェアが必要です

3-4　MS-DOS マクロアセンブラ

3-4-1　マクロアセンブラの特徴

この項では，MS-DOS のマクロアセンブラ［MASM］の特徴について簡単に解説します．

MASM によるプログラミングはアセンブリ言語である以上，高級言語とは言えませんが，モニタによるマシン語プログラミングなどと比べると，マクロ機能や疑似命令と呼ばれるアセンブラ自体を制御する命令を使うことによって，高級言語に近いプログラミングを行うことが可能です．

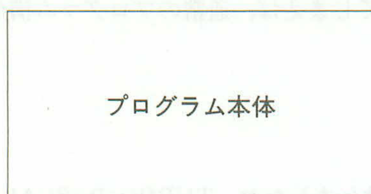
また，アセンブルされたオブジェクト・ファイルをリンカによって BASIC，C，Pascal などの高級言語とリンクすることも容易にできます．

3-4-1-1　マクロ機能について

マクロ機能とは，プログラミングしていく過程で同じ処理が頻出するような場合，その処理をマクロ定義によってサブルーチン化することです．そして，その処理が必要なところでマクロ名でマクロ・コールします．

一般的なマクロ定義の書式を次に示します．

［マクロ名］　macro　［仮引数,・・・・］



endm

マクロ・コールの書式は次のようになります．

マクロ名　［実引数,・・・・］

マクロ・コールとサブルーチン・コールの違いは，サブルーチン・コールはcall命令によってサブルーチンのある場所にジャンプし，ret命令によってリターンするという動作をします．一方，マクロ・コールはコールされる都度，その場所にプログラム本体が挿入されるため，ジャンプ，リターンは行いません．そのため実行時間は短縮されますが，生成された機械語コードは重複した部分ができるので，メモリを余分に消費します．

3-4-1-2 演算子について

MASMで使われる演算子には表3-4-1に示したようなものがあります．

演算子	機能	優先順位
オーバーライド演算子		
PTR	オーバーライドのタイプ，距離を明示	3
:	セグメント・オーバーライド	2
SHORT	ラベルまでの距離が+−127バイト以下であることを示す	11
THIS	距離，タイプを明示したオペランドの作成	3
HIGH／LOW	16ビット絶対値，アドレス値の上位／下位8ビットを分離	4
値を返す演算子		
SEG	変数，ラベルを囲んでいるセグメントのセグメント・ベースの値を返す	3
OFFSET	変数，ラベルのオフセット値を返す	3
TYPE	オペランドの距離，タイプを返す	3
.TYPE	変数のモード，特性を返す	11
LENGTH	変数のタイプの単位の数を返す	1
SIZE	変数に割り振られたバイトの合計数を返す	1
レコード指定演算子		
シフトカウント	レコードのLOW ENDからフィールドのLOW ENDまでのビット数	1
MASK	フィールドをマスクした時の最大のレコード値	1
WIDTH	フィールド，レコードに含まれるビット幅の数	1
算術演算子		
*／	乗算　除算	5
MOD	剰余	5
SHR	右シフト	5
SHL	左シフト	5
−	負符号	6
+−	加算　減算	6
関係演算子		
EQ／NE	等しい／等しくない	7
LT／LE	より小さい／より小さいか等しい	7
GY／GE	より大きい／より大きいか等しい	7
論理演算子		
NOT	否定	8
AND	論理積	9
OR	論理和	10
XOR	排他的論理和	10
マクロ演算子		
&	マクロ本体中の仮引数が' '内にある場合，それが仮引数であることを明示する	
<>	<>内を単一のパラメータとして扱う	
%	%に続く式を数値に変換する	
!	!に続くキャラクタをその文字通りに扱う	
;;	;;に続くコメントはマクロ展開されない	

表3-4-1　MASMで使用される演算子一覧

3-4-1-3 疑似命令について

MASMで使われる疑似命令には表3-4-2に示したようなものがあります。

メモリ疑似命令

疑似命令	機能
ASSUME	どのレジスタを使ってセグメントを参照するかを仮定する
COMMENT	任意の大きさのコメント・ブロックを挿入する
DB／DW／DD／DQ／DT	バイト／ワード／ダブルワード／クォドワード／テンバイト単位でデータを定義する
END	プログラムの終了と実行開始アドレスの指定
EQU	式の値を名前に割り当てる
=	式の値を名前に割り当てる(再定義可能)
EVEN	プログラム・カウンタ(PC)を偶数番地の境界に進める
EXTRN	他のモジュールで定義されている名前であるという宣言
GROUP	複数のセグメントを一つの名前で参照できるようにグループ化する
INCLUDE	他のソース・ファイルを現ソース・ファイルに挿入する
LABEL	名前にタイプを割り当てる
NAME	モジュール名の定義
ORG	ロケーション・カウンタのセット
PROC～ENDP	プロシージャの定義
PUBLIC	他のモジュールから参照される名前であるという宣言
.RADIX	インプット・ベース(基数)の変更
RECORD	フィールド内のビット数の指定
SEGMENT～ENDS	セグメントの定義
STRUC～ENDS	構造体型の型定義

条件疑似命令

IF～ELSE～ENDIF	条件が真ならIFブロックを，偽ならELSEブロックをアセンブルする
IF／IFE	式が，0でない／0である，を判断する
IF1／IF2	アセンブラが，パス1／パス2，のどちらを実行中か判断する
IFDEF／IFNDEF	シンボルが，定義されている／定義されていない，を判断する
IFB／IFNB	引数が，ブランクである／ブランクでない，を判断する
IFIDN／IFDIF	2つの引数文字列が，同じである／同じでない，を判断する

マクロ疑似命令

MACRO～ENDM	マクロの定義
EXITM	マクロからの抜け出し
LOCAL	マクロ内での局所的シンボルであるという宣言
PURGE	マクロ定義の削除

リピート疑似命令

REPT～ENDM	繰り返し
IRP～ENDM	不定回数の繰り返し
IRPC～ENDM	不定回数の文字繰り返し

リスティング疑似命令

PAGE	ページ・サイズの指定，ページの開始
TITLE	タイトルの指定
SUBTTL	サブタイトルの指定
%OUT	アセンブル中にテキストを端末に表示する
.LIST／.XLIST	すべての行をコードとともにリストする／リストしない
.SFCOND／.LFCOND／.TFCOND	偽条件式を含むリスティングを出力しない／出力する／現在のセッティングを反転する
.XALL／.LALL／.SALL	マクロによって生成されたソース・コードをリストする／すべてのマクロを展開しリストする／すべてのリスティングを抑止する
.CREF／.XCREF	クロスリファレンスの作成を行う／抑止する

表3-4-2 MASMで使用される疑似命令一覧

3-4-2　MS-DOS のシステム・コールの利用法

機械語でプログラミングを行う場合，コンソール，プリンタ，ディスクドライブ等のデバイスをアクセスするルーチンをユーザーがI／Oポートを使って自作するのは，非常に開発効率の悪いことです．MS-DOS では，これらのデバイス・アクセスのサブルーチンを MSDOS. SYS 内に組み込んであり，ユーザーは INT 21 H のソフトウェア割り込みを行うことで，これらのサブルーチンを使用することができます．これをシステム・コール(ファンクション・コール)と呼びます．

MS-DOS ver 2.11 では 72 種類のシステム・コールが用意されており，ver 3.0 で付加されたものと共に付録にその一覧表を示します．

ここでは，主なシステム・コールについて機能別に分けて解説します．

INT 21 H によるシステム・コールは，AH レジスタにパラメータを設定して行います．このパラメータをファンクション・リクエスト番号と呼びます．

3-4-2-1　コンソール入出力に関するシステム・コール

ファンクション・リクエスト番号 01 H，02 H，06 H～0 CH のシステム・コールはコンソール(キーボード，ディスプレイ)との入出力を行います．

01 H　コンソールからの 1 文字入力

内部ループ型で，入力された文字のアスキー・コードを AL レジスタに格納します．また，入力された文字をエコーバックし，＾C をチェックします．

02 H　コンソールへの 1 文字出力

DL レジスタにセットされた文字キャラクタをコンソールに出力します．＾C をチェックします．

06 H　コンソールへの 1 文字出力

DL レジスタにセットされた値が FFH であった場合，入力モードになります．その他の値であれば，出力モードになります．入力モードは非ループ型で，エコーバックはありません．また，＾C をチェックしません．

07 H　コンソールからの 1 文字入力

内部ループ型で，入力された文字のアスキー・コードを AL レジスタに格納します．入力された文字のエコーバックはなく，＾C もチェックしません．

08 H　コンソールからの 1 文字入力

内部ループ型で，入力された文字のアスキー・コードを AL レジスタに格納します．入力された文字のエコーバックはなく，＾C をチェックします．

09 H　コンソールへの文字列出力

DX レジスタにセットされたオフセット・アドレスの文字列データをコンソールに出力します．文字列の最後は〝$〃でなければなりません．

0AH　コンソールからの文字列入力

コンソールから入力された文字列をDXレジスタにセットされたオフセット・アドレス+2番地から格納します．入力の終わりはリターン(0 DH)です．入力バッファの先頭バイトには，入力可能なバッファサイズを，ユーザーがあらかじめセットしておく必要があります．入力バッファの先頭+1バイトには，リターンコードを含まない実際の入力文字数が格納されます．入力可能な最大文字数は，リターンコードを含めて255文字までです．また，テンプレートや漢字変換機能を利用することもできます．＾Cをチェックします．

0 BH　キーボード・バッファのステータス検査

キーボード・バッファに文字が入っている場合には，ALレジスタにFFHが，入っていない場合には，0がセットされます．

0CH　コンソールからの文字列入力

キーボード・バッファを空にした後，ALレジスタにセットされたファンクション・リクエスト番号(01 H，06 H，07 H，08 H，0 AH)に対応した文字入力を行います．

3-4-2-2　外部入出力装置に関するシステム・コール

ファンクション・リクエスト番号03 H，04 H，05 Hのシステム・コールはプリンタ，補助入出力装置(RS-232 Cであることが多い)との入出力を行います．

03 H　補助装置からの1文字入力

ALレジスタに補助装置から入力された文字のアスキー・コードが格納されます．

04 H　補助装置への1文字出力

DLレジスタにセットされた文字キャラクタを補助装置に出力します．

05 H　プリンタへの1文字出力

DLレジスタにセットされた文字キャラクタをプリンタに出力します．

3-4-2-3　FCBによるファイル・アクセスに関するシステム・コール

ファンクション・リクエスト番号0 DH～24 H，27 H～29 H，2 FHのシステム・コールはFCBによるファイル・アクセスを行います．これらのシステム・コールは旧バージョンとの互換を保つために用意されたもので，階層ディレクトリをサポートしていません．互換性を考慮しなくてよい場合は，後述の「ファイル・ハンドルによるファイル・アクセスに関するシステム・コール」を利用した方が良いでしょう．

ここでは詳しい解説は省きます．入力条件，リターン情報に関しては，一覧表を参照してください．

3-4-2-4 ファイル・ハンドルによるファイル・アクセスに関するシステム・コール

ファンクション・リクエスト番号3CH～46H，4EH，4FH，56H，57Hのシステム・コールはファイル・ハンドルによるファイル・アクセスを行います．これらのシステム・コールを使ってファイルのオープン処理を行うことにより，AXレジスタにファイル・ハンドル番号という1ワードの情報が返されます．以後，このファイルに対するアクセスは，このファイル・ハンドル番号を使用して行われます．

これらのシステム・コールを利用することによって，ユーザーはプログラム中で，MS-DOSの大きな特徴のひとつである階層ディレクトリをプログラム中で扱うことができます．

3CH 新規ファイルの作成

このシステム・コールは，DXレジスタにパス名を指定したファイルの格納されているアドレスを，CXレジスタにそのファイルの属性を設定して行います．ファイルのオープン・モードは自動的にリード／ライト・モードに設定され，AXレジスタにファイル・ハンドル番号が返されます．エラー発生時にはキャリーフラグがセットされ，AXレジスタにエラー・コードが返されます．ファイルの属性は表3-4-3を参照してください．また，エラー・コードは表3-4-4を参照してください．

番 号	ファイル属性
01H	リード・オンリー・ファイル
02H	隠しファイル
04H	システム・ファイル
08H	ボリュームラベル
10H	ディレクトリ
20H	保存ビット

表3-4-3 ディスク・ファイルの属性

コード	意 味
01H	無効なファンクション・コード(Invalid function code)
02H	ファイル名が見つからない(File not found)
03H	パス名が見つからない(Path not found)
04H	オープン・ファイルが多過ぎる(Too many open handoles left)
05H	アクセスできない(Access denied)
06H	無効なハンドル(Invalid handole)
07H	メモリ・コントロール・ブロックが破損している(Memory controll blocks destroyed)
08H	メモリが不足している(Insufficient memory)
09H	無効なメモリ・ブロック・アドレス(Invalid memory block adress)
0AH	無効な環境(Invalid environment)
0BH	無効な書式(Invalid format)
0CH	無効なアクセス・コード(Invalid access code)
0DH	無効なデータ(Invalid data)
0EH	未使用(RESERVED)
0FH	無効なドライブ名(Invalid drive)
10H	カレント・ディレクトリを削除しようとした(Attempt to remove the current directry)
11H	同じデバイスはない(Not same device)
12H	これ以上ファイルはない(Not more file)
13H	ディスクがライト・プロテクトされている(Disk is write protected)

14H	ディスク・ユニット不良(Bad disk unit)
15H	ディスクの準備ができていない(Drive not ready)
16H	無効なディスク・コマンド(Invalid disk command)
17H	CRCエラー(CRC error)
18H	無効な長さ(Invalid length)
19H	シーク・エラー(Seek error)
1AH	MS-DOSのディスクではない(Not an MS-DOS disk)
1BH	セクタが見つからない(Sector not found)
1CH	紙切れ(Out of paper)
1DH	書き込み失敗(Write fault)
1EH	読み出し失敗(Read fault)
1FH	通常の失敗(General failure)
20H	シェアリング違反(Shareing violation)
21H	ロック違反(Lock violation)
22H	ディスク指定の失敗(Wrong disk)
23H	FCBが使用できない(FCB unavailable)
24H | 31H	未使用(RESERVED)
32H	ネットワーク・リクエストがサポートされていない(Network request not supported)
33H	リモート・コンピュータがLISTENしていない(Remote computer not listening)
34H	ネットワーク名が二重定義されている(Duplicate name on netw)
35H	ネットワーク名が見つからない(Network name not found)
36H	ネットワーク・ビジー(Network busy)
37H	ネットワーク・デバイスはこれ以上ない(Network device no longer exists)
38H	ネットワークBIOSの限界を越えた(Net BIOS command limit exceeded)
39H	ネットワーク・アダプターのハード・エラー(Network adapter hardware error)
3AH	ネットワークからの不当な応答(Incorrect response from network)
3BH	予期できないネットワーク・エラー(Unexpected network error)
3CH	リモート・アダプターが合致しない(Incompatible remote adapt)
3DH	プリント待ち行列が一杯(Print queue full)
3EH	待ち行列は一杯ではない(Quene not full)
3FH	プリント・ファイルのスペースが十分にない(Not enough space for print file)
40H	ネットワーク名は削除されている(Network name was delected)
41H	アクセスできない(Access denied)
42H	ネットワーク・デバイスのタイプが不適当(Network device type incorrect)
43H	ネットワーク名が見つからない(Network name not found)
44H	ネットワーク名の限界を越えた(Network name limit exceeded)
45H	ネットワークBIOSセッションの限界を越えた(Net BIOS session limit exceeded)
46H	一時休止(Temporamily paused)
47H	ネットワーク・リクエストが拒否された(Network request not accepted)
48H	プリンタ，ディスクのリディレクション休止(print or disk redirection is paused)
49H | 4FH	未使用(RESERVED)
50H	ファイルが存在する(File exists)
51H	未使用(RESERVED)
52H	作成不能(Cannot make)
53H	割り込みタイプ24Hの失敗(Interrupt 24H fault)
54H	ストラクチャの不良(Out of structure)
55H	割り当て済み(Already assigned)
56H	無効なパスワード(Invalid password)
57H	無効なパラメータ(Invalid parameter)
58H	ネットワークへの書き込み失敗(Net Write fault)

表3-4-4 システム・コールのエラー・コード

3DH ファイルのオープン

既存のファイルのオープン処理を行います．DX レジスタにパス名を指定したファイルの格納されているアドレスを設定して行います．既存のファイルが対象ですので，ファイル属性の設定は必要ありませんが，ファイルのオープン・モードを AL レジスタに設定します．オープン・モード設定は，

00 H リード・モード
01 H ライト・モード
02 H リード／ライト・モード

となっています．このシステム・コールを行うことによって，AX レジスタにファイル・ハンドル番号が返されます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

3EH ファイルのクローズ

オープンされているファイルをクローズします．BX レジスタにクローズするファイルのファイル・ハンドル番号を設定して行います．このシステム・コールを行うことによって，バッファに残っていたデータのディスクへの書き込みとファイル・ハンドルの開放が行われます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

3FH データの読み込み

リード・モードでオープンされたファイルの読み込みを行います．BX レジスタにファイル・ハンドル番号，DS／DX にバッファのアドレス，CX レジスタに読み込むバイト数を設定します．AX レジスタに読み込んだバイト数が返されます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

40 H データの書き込み

ライト・モードでオープンされたファイルへの書き込みを行います．BX レジスタにファイル・ハンドル番号，DS／DX にバッファのアドレス，CX レジスタに書き込むバイト数を設定します．AX レジスタに書き込んだバイト数が返されます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

41 H ファイルの削除

DS／DX で示されたファイルを削除します．DX レジスタにファイル名の置かれているアドレスを設定します．ファイル名にパス名を指定することはできますが，ワイルド・キャラクタを使用することはできません．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

42 H　ファイル・ポインタの移動

AL レジスタに示される始点から，CX，DX レジスタに示される移動量だけファイル・ポインタを移動します．AL レジスタに設定する移動の始点は，

00 H　ファイルの先頭

01 H　現在の位置

02 H　ファイルの終わり

となっています．CX レジスタには移動量の上位2バイトを，DX レジスタには下位2バイトを設定しますが，その数値は符号付4バイトの数値となります．BX レジスタにはファイル・ハンドル番号を設定して行います．AX レジスタに新規のポインタ・ロケーションが返されます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

43 H　ファイル属性の変更

現在のファイル属性のチェックと新しいファイル属性の設定を行います．AL レジスタに 00 H を設定した場合には，属性のチェック，01 H を設定した場合には属性の設定になります．変更可能なファイル属性は以下に示す4つで，CX レジスタに設定する値によって決まります．

CX：01 H　　リード・オンリー・ファイル

CX：02 H　　隠しファイル

CX：04 H　　システム・ファイル

CX：20 H　　通常ファイル

DX レジスタにファイル名の置かれているアドレスを設定します．ファイル名にパス名を指定することはできますが，ワイルド・キャラクタを使用することはできません．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

44 H　デバイスに対する I／O コントロール

AL レジスタにコントロール・コードを設定して行います．コントロール・コードの種類と機能を表3-4-5に示します．

コード	機　　能
00H	コントロール・データの取得
01H	コントロール・データのセット
02H	コントロール・データを転送する
03H	コントロール・データを受け取る
04H	コントロール・データを転送する
05H	コントロール・データを受け取る
06H	入力ステータスの取得
07H	出力ステータスの取得

表3-4-5　I／O コントロール・コード

00 H　デバイス・コントロール・データの取得

BX レジスタにファイル・ハンドル番号を設定して行います．DX レジスタに 2 バイトのデバイス・データが返されます．デバイス・データの 2 バイトの意味を表 3-4-6 に示します．エラー発生時にはキャリー・フラグをセットし，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

（ビット 7 が 1 の場合）

Bit	値	意　味
0	1	コンソール入力
1	1	コンソール出力
2	1	NULLデヴァイス
3	1	クロック・デヴァイス
4		予備
5	0	文字のチェックを行う（C，P，S，Z）
	1	文字のチェックを行わない
6	0	EOFを入力する
7	1	
8 | 13		予備
14	1	ストリングのコントロールを処理する
15		予備

（ビット 7 が 0 の場合）

Bit	値	意　味
0 | 5		予備
6	0	書き込まれたファイル
7	0	
8 | 15		予備

表 3-4-6　デバイス・データの意味

01 H　デバイス・コントロール・データのセット

BX レジスタにファイル・ハンドル番号を，DH レジスタに 00 H を設定して行います．DX レジスタに 2 バイトのデバイス・データが返されます．デバイス・データは表 3-4-6 を参照してください．エラー発生時にはキャリー・フラグをセットし，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

02 H　デバイス・コントロール・データをキャラクタ・デバイスに送る

BX レジスタにキャラクタ・デバイスのファイル・ハンドル番号を，CX レジスタに転送するコントロール・データのバイト数を，DX レジスタにデータ・バッファのオフセット・アドレスを設定して行います．デバイス・ドライバは IOCTL インターフェースをサポートしている必要があります．AX レジスタに転送されたデータのバイト数が返されます．エラー発生時にはキャリ

ー・フラグをセットし，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

03H　コントロール・データをキャラクタ・デバイスから受け取る

BXレジスタにキャラクタ・デバイスのファイル・ハンドル番号を，CXレジスタに読み取るコントロール・データのバイト数を，DXレジスタにデータ・バッファのオフセット・アドレスを設定して行います．デバイス・ドライバはIOCTLインターフェースをサポートしている必要があります．AXレジスタに読み取ったデータのバイト数が返されます．エラー発生時にはキャリー・フラグをセットし，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

04H　コントロール・データをブロック・デバイスに送る

BLレジスタにドライブ番号(00H＝デフォルト,01H＝A)…を，CXレジスタに転送するコントロール・データのバイト数を，DXレジスタにデータ・バッファのオフセット・アドレスを設定して行います．デバイス・ドライバはIOCTLインターフェースをサポートしている必要があります．AXレジスタに転送されたデータのバイト数が返されます．エラー発生時にはキャリー・フラグをセットし，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

05H　コントロール・データをブロック・デバイスから読み取る

BLレジスタにドライブ番号(00H＝デフォルト,01H＝A…)を，CXレジスタに読み取るコントロール・データのバイト数を，DXレジスタにデータ・バッファのオフセット・アドレスを設定して行います．デバイス・ドライバはIOCTLインターフェースをサポートしている必要があります．AXレジスタに転送されたデータのバイト数が返されます．エラー発生時にはキャリー・フラグをセットし，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

06H　ファイル・ハンドルの状態のチェック

BXレジスタにファイル・ハンドル番号を設定して行います．ALレジスタに返された値とステータスの意味を下に示します．

＜値＞	＜デバイス＞	＜入力ファイル＞
00H	非レディ状態	ポインタがEOFを指している
FFH	レディ状態	レディ状態

エラー発生時にはキャリー・フラグをセットし，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

07H　ファイル・ハンドルの状態のチェック

BXレジスタにファイル・ハンドル番号を設定して行います．ALレジスタに返された値とステータスの意味を下に示します．

＜値＞　　＜デバイス＞　　＜出力ファイル＞

00 H　　非レディ状態　　レディ状態

FFH　　レディ状態　　レディ状態

出力ファイルはディスク・フルでもレディ状態の値を返します．エラー発生時にはキャリー・フラグをセットし，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

45 H　ファイル・ハンドルのコピー

既にオープンされているファイル・ハンドルを取り出し，同じファイルを扱う新規のファイル・ハンドルを作成します．BX レジスタにファイル・ハンドル番号を設定して行います．AX レジスタに新規に作成されたファイル・ハンドル番号が返されます．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

46 H　ファイル・ハンドルのコピー

既にオープンされているファイル・ハンドルを取り出し，同じ位置の同じファイルを示す新規のファイル・ハンドルを作成します．BX レジスタにファイル・ハンドル番号を設定します．CX レジスタにコピー先のファイル・ハンドル番号を設定します．エラー発生時にはキャリーフラグがセットされ，AX レジスタにエラー・コードが返されます．エラー・コードは表 3-4-4 を参照してください．

4EH　ファイル名の検索

DS：DX に示されるファイル名を検索します．DX レジスタにファイル名の置かれているアドレスを設定します．ファイル名にパス名を指定することができます．ワイルド・キャラクタを使用することもできますが，パス名には使用できません．CX レジスタに検索するファイルの属性を設定します．指定されたファイルが見つかった場合には，ディスク・バッファ内に図 3-4-1 に示すような情報がセットされます．

0 1 2 3 4 5 6 7 8 9 A B C D E F

0: ファイル名　拡張子　予備　ドライブ番号　ファイル属性

1: 予備　時刻　日付　大きさ　パック・ネーム　属性

2: パック・ネーム

図 3-4-1　ディスク・ファイル情報

このシステム・コールによって得られた情報のアドレスはファンクション・リクエスト番号2FHのシステム・コールによって得ることができます．エラー発生時にはキャリーフラグがセットされ，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

4FH　ファイル名の検索

4EHのシステム・コールによって得られた情報をもとにファイル名の検索を行います．パラメータの指定は必要ありません．

56H　ディレクトリ・エントリの移動

DS／DXに示されるパス名，ファイル名をES／DIに示されるパス名，ファイル名に変換します．DXレジスタに既存のファイル名の置かれているアドレスを設定します．DIレジスタに移動先のファイル名の置かれているアドレスを設定します．ファイル名にパス名を指定することができます．ワイルド・キャラクタを使用することはできません．エラー発生時にはキャリーフラグがセットされ，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

57H　ファイルの時間情報の取得，セット

ファイルの時間情報(日付／時刻)のチェックと設定を行います．ALレジスタに00Hを設定した場合にはチェック，01Hを設定した場合には設定になります．BXレジスタにファイル・ハンドル番号を設定します．ALレジスタの値が01Hのとき，CXレジスタに設定する時刻を，DXレジスタに日付を設定します．エラー発生時にはキャリーフラグがセットされ，AXレジスタにエラー・コードが返されます．エラー・コードは表3-4-4を参照してください．

3-4-2-5　プログラムの実行に関するシステム・コール

ファンクション・リクエスト番号00H，31H，4BH～4DHのシステム・コールはプログラムの実行，終了などを行います．ここでは詳しい解説はしません．入力条件，リターン情報に関しては，一覧表を参照してください．

00H　プログラムの終了

実行中のプログラムの使用領域を開放して制御を親プロセスに戻します．SレジスタにはPSPのセグメント・アドレスを設定しておかなくてはなりません．

31H　プログラムの終了

実行中のプログラム(またはその一部)をメモリに常駐したまま制御を親プロセスに戻します．

4BH　プログラムのロード＆実行

プログラム中からチャイルド・プロセスを起動します．

4CH　プログラムの終了

実行中のプログラムの使用領域を開放して制御を親プロセスに戻します．最も一般的なプログ

ラム終了方法で，オプションの機能としてALレジスタにリターン・コードを設定することにより親プロセスにパラメータを渡すことができます．

4DH　チャイルド・プロセスからのリターン・コードを受け取る

4CHのシステム・コールで設定されたリターン・コードを読み出します．

3-4-2-6　メモリ管理に関するシステム・コール

ファンクション・リクエスト番号48H～4AHのシステム・コールはメモリ領域のコントロールを行います．ここでは機能を述べるだけにとどめ，詳しい解説は省きます．入力条件，リターン情報に関しては，一覧表を参照してください．

48H　メモリ領域の割り当て

49H　メモリ領域の開放

4AH　メモリ領域のサイズ変更

3-4-2-7　ディレクトリ管理に関するシステム・コール

ファンクション・リクエスト番号39H～3BH，47Hのシステム・コールは階層ディレクトリの管理を行うものです．これらのシステム・コールはCOMMAND．COMのビルトイン・コマンドとして用意されています．ここでは機能を述べるだけにとどめ，詳しい解説は省きます．入力条件，リターン情報に関しては，一覧表を参照してください．

39H　サブディレクトリの作成

3AH　サブディレクトリの削除

3BH　カレントディレクトリの移動

47H　カレントディレクトリの表示

3-4-2-8　その他のシステム・コール

ファンクション・リクエスト番号25H，2AH～2EH，30H，33H～38H，54Hのシステム・コールのうち，ほとんどはCOMMAND．COMのビルトイン・コマンドとして用意されています．ここでは機能を述べるだけにとどめ，詳しい解説は省きます．入力条件，リターン情報に関しては，一覧表を参照してください．

25H　割り込みベクトルのセット

2AH　日付情報のチェック

2BH　日付のセット

2CH　時刻情報のチェック

2DH　時刻のセット

2EH　ベリファイ・フラグのセット，リセット

30H　DOSのヴァージョン番号のチェック

33H　^Cのチェックのセット，リセット

35H　割り込みベクトルのチェック

36H　ディスクの空き領域のチェック

38H　国別情報のチェック

3-4-3　その他のソフトウェア割り込み

MS-DOSは20H～3FHまでの32個の内部割り込み用ジャンプ・テーブルを使用しており，このうち20H～27Hの8個をユーザーに開放しています．この8個の内部割り込みによって実行される各サブルーチンの機能を表3-4-7に示します．

ユーザーはINT命令を用いることによって，これらのサブルーチンをプログラム中で使用することができます．

番号	機　能
20H	プログラムの終了
21H	ファンクション・コール(システム・コール)
22H	プログラム終了アドレス
23H	C処理
24H	致命的エラー処理
25H	アブソリュート・ディスク・リード
26H	アブソリュート・ディスク・ライト
27H	プログラムの終了

表3-4-7　MS-DOSのソフトウェア割り込み

20H　プログラムの終了

使用していたプログラム・エリアを開放してプログラムを終了させるもので，一般的なプログラム終了方法です．ただし，このソフトウェア割り込みは，CSレジスタがPSPのセグメント・アドレスを示していることが条件ですので，EXEメモリ・モデルのプログラムで使用することはできません．EXEメモリ・モデルのプログラムを終了させる場合は，リクエスト番号4CHのシステム・コールを使用するのが良いでしょう．

21H　ファンクション・コール(システム・コール)

ファンクション・リクエストです．

22H，23H，24H

これら3つの割り込みテーブルは，INT命令を使ってプログラム中で使用するものではありません．22Hのテーブルは，プログラム終了アドレスが格納されています．MS-DOS上でプログラムのロード&実行を行った場合，そのプログラムの終了時に制御を戻すアドレスが設定されます．23Hのテーブルは，^Cに対する処理ルーチンのアドレスが格納されています．

MSDOS.SYSが^Cを検出した場合，このテーブルを参照して制御を移します．このテーブルを応用して，ユーザーがプログラム中で独自の^Cに対する処理ルーチンを作成することも可能です．24 Hのテーブルは，致命的エラーに対する処理ルーチンのアドレスが格納されています．致命的エラーが発生した場合，MSDOS.SYSはこのテーブルを参照して制御を移します．

25 H　アブソリュート・ディスク・リード

この割り込みは，論理セクタ単位でディスクからデータを読み込みます．入力条件を示します．

AL：ドライブ番号(0，1……)

DS：データ・バッファのセグメント・ベース

BX：データ・バッファのオフセット・アドレス

CX：読み込みセクタ数

DX：読み込み開始セクタ(論理セクタ番号)

エラー発生時には，キャリー・フラグをセットし，ALレジスタにエラー・コードを返します．この割り込み処理を実行することによって，セグメント・レジスタ以外のレジスタは全て破壊されるので，注意が必要です．また，全てのフラグ・レジスタを自動的にスタックするので，実行後にPOPFを行う必要があります．

26 H　アブソリュート・ディスク・ライト

この割り込みは，論理セクタ単位でディスクにデータを書き込みます．入力条件を示します．

AL：ドライブ番号(0，1……)

DS：データ・バッファのセグメント・ベース

BX：データ・バッファのオフセット・アドレス

CX：書き込みセクタ数

DX：書き込み開始セクタ(論理セクタ番号)

エラー発生時には，キャリー・フラグをセットし，ALレジスタにエラー・コードを返します．この割り込み処理を実行することによって，セグメント・レジスタ以外のレジスタは全て破壊されるので，注意が必要です．また，全てのフラグ・レジスタを自動的にスタックするので，実行後にPOPFを行う必要があります．

27 H　プログラムの終了

特殊なプログラム終了方法で，実行していたプログラムをメモリに常駐したまま親プロセスに制御を戻します．このソフトウェア割り込みは，CSレジスタがPSPのセグメント・アドレスを示していることと，DXレジスタがプログラムの最終アドレスを示していることが条件ですので，前述の20 H同様，EXEメモリ・モデルのプログラムで使用することはできません．リクエスト番号31 Hのシステム・コールも同じ機能を持ちます．

3-4-4　MS-DOS ver 3.1で追加されたシステム・コール

MS-DOS ver 3.1で新たに付加(または変更)されたシステム・コールについて解説します.

1BH　カレント・ドライブのデータの取得

AHレジスタ以外の入力パラメータは必要ありません. ALレジスタに1クラスタあたりのセクタ数が, CXレジスタに1セクタあたりのバイト数が, DXレジスタに1ドライブあたりのクラスタ数が返されます. また, DS／BXにFAT-IDのオフセット・アドレスが返されます.

1CH　ドライブのデータの取得

DLレジスタにドライブ番号(00H＝デフォルト,01H＝A…)を設定して行います. ALレジスタにFFHが返された場合はドライブの指定が無効ということです. FFH以外の値が返された場合は1クラスタあたりのセクタ数です. CXレジスタに1セクタあたりのバイト数が, DXレジスタに1ドライブあたりのクラスタ数が, DS／BXにFAT-IDのオフセット・アドレスが返されます. このシステム・コールはDLレジスタに00Hを設定した場合, リクエスト番号1BHと同じ機能を持ちます. また, DS／BXにFAT-IDが返される点を除けば, リクエスト番号36Hと同じ機能を持ちます.

44H　デバイスに対するI／Oコントロール

ALレジスタに設定するコントロール・コードが4種類追加されています.

08H　IOCTLの交換性のチェック

BLレジスタにドライブ番号をセットして行います. AXレジスタに返される値が00Hの場合は変換可能, 01Hの場合は変換不可能です. 変換可能なドライブとは通常のディスク・ドライブ, 変換不可能なドライブはハード・ディスクやRAMディスクのようなものです. エラー発生時にはキャリーフラグをセットし, AXレジスタにエラー・コードが返されます.

09H　ドライブのリダイレクトのチェック

BLレジスタに設定されたドライブがローカル(MS-Networksのワーク・ステーション)であるか, リモートであるかをチェックします. BLレジスタにドライブ番号を設定して行います. DXレジスタに返される2バイトのデータのビット12が0の場合はローカル・ドライブ, 1の場合はリモート・ドライブです. エラー発生時にはキャリーフラグをセットし, AXレジスタにエラー・コードが返されます.

0AH　ファイルのリダイレクトのチェック

BXレジスタで示されたファイルがローカルであるか, リモートであるかをチェックします. BXレジスタにファイル・ハンドル番号を設定して行います. DXレジスタに返される2バイトのデータのビット15が0の場合はローカル, 1の場合はリモート・ファイルかデバイスです. エラー発生時にはキャリーフラグをセットし, AXレジスタにエラー・コードを返します.

0BH　リトライ回数の設定

MS-DOSがディスク・アクセスに失敗した時のリトライの回数を設定します. BXレジスタに

リトライの回数，CXレジスタにリトライの間隔の時間を設定して行います．なお，MS-DOS起動時のリトライ回数のデフォルト値は3回です．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

26H 新しいPSPを作成する

DXレジスタに新しいPSPのセグメント・アドレスを設定して行います．このシステム・コールは旧バージョンとの互換性を保つために用意されているもので，互換性を考慮しなくてもよい場合にはリクエスト番号4BHのシステム・コールを使ってチャイルド・プロセスの起動を行ったほうが良いでしょう．

38H 国別情報のチェック／セット

チェックに関してはver 2.11と同じです．ここでは詳しい解説はしません．一覧表を参照してください．

3DH ファイルのオープン

システム・ファイル，隠しファイルを含めたあらゆるファイルを入力，または出力モードでオープンします．ALレジスタにファイル・アクセス制御コードをビット単位で設定します．また，DS：DXにオープンするファイルのパス名のアドレスを設定して行います．



図3-4-2 ALレジスタのパラメータ

●インヘリッド・ビット

リクエスト番号4BHによって起動されたチャイルド・プロセスから情報を渡されるかどうかを指定します．0の場合には情報が渡されます．1の場合は渡されません．

●シェアリング・モード

すでにオープンされているファイルをアクセスできるかどうかを指定します．ただし，SHARE.EXEがインストールされていない場合にはver 2.0と同様な処理が行われます．

●アクセス・コード

ファイルにどのようなアクセスができるかを指定します．

シェアリング・モードとアクセス・コードの意味を表3-4-8に示します．

値	シェアリング・モード	意　味
000	コンパチブル	どんなプロセスでもこのモードなら何度でもオープン可能
001	リード／ライト不可	コンパチブル・モードでのオープン，リード，ライト・アクセスはできない
010	ライト不可	コンパチブル・モードでのオープン，ライト・アクセスはできない
011	リード不可	コンパチブル・モードでのオープン，リード・アクセスはできない
100	不可なし	コンパチブル・モードでのオープンはできない
値	**アクセス・コード**	**意　味**
0000	リード	リード不可，リード／ライト不可のシェアリング・モードでオープンできない
0001	ライト	ライト不可，リード／ライト不可のシェアリング・モードでオープンできない
0010	リード／ライト	リード不可，ライト不可，リード／ライト不可のシェアリング・モードでオープンできない

表3-4-8　シェアリング・モード／アクセス・コードの意味

58H　アロケーション・ストラテジの取得／設定

ALレジスタに00Hをセットして行った場合にはAXレジスタにストラテジが返されます．01Hをセットした場合にはBXレジスタにセットされたストラテジが設定されます．アロケーション・ストラテジの意味は次のとおりです．

00H：下位　　最も下位の利用可能なブロックを割り当てます．

01H：最小　　必要最小限の利用可能なブロックを割り当てます．

02H：上位　　最も上位の利用可能なブロックを割り当てます．

エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

59H　拡張されたエラー・コードの取得

エラーに対するより詳細な情報を取得します．AXレジスタに拡張されたエラー・コード，BHレジスタにエラー・クラス，BLレジスタに可能な対処，CHレジスタにローカスが返されます．このシステム・コールの実行によりCL，DX，SI，DI，BP，DS，ESの各レジスタは破壊されます．

＜エラー・クラス＞

01H：メモリ容量やI／Oチャネルなどの資源の不足

02H：エラーではないが終了した方がよい状況が発生している

03H：権限の問題

04H：システム・ソフトウェアの内部エラー

05H：ハードウェアに起因するエラー

06H：現在のプロセスが原因でないシステム・ソフトウェアのエラー

07H：アプリケーション・プログラムのエラー

08 H：ファイルが存在しない

09 H：無効なフォーマット

0AH：ファイルが内部的にロックされている

0BH：ディスクの不良

0CH：その他の原因によるエラー

＜可能な対処＞

01 H：ユーザーに確認を求めて再試行

02 H：休止後に再試行

03 H：ユーザーに再度の入力要求

04 H：メモリをクリアして終了

05 H：プログラムを終了

06 H：エラー・コードに対応した処理

07 H：ユーザーがディスク交換等の動作を行う必要がある

＜ローカス・エラーの発生箇所＞

01 H：不明

02 H：ディスク・ドライブ等のランダム・アクセス・ブロック・デバイスで発生したエラー

03 H：ネットワークで発生したエラー

04 H：プリンタ等のシリアス・アクセス・キャラクタ・デバイスで発生したエラー

05 H：RAMで発生したエラー

5AH　一時ファイルの作成

CXレジスタに作成するファイルの属性，DS／DXにパス名を設定して行います．AXレジスタにファイル・ハンドルが返されます．DXレジスタに示されるパス名の後には00 Hで始まる13バイトのメモリ領域が必要です．MS-DOSはディレクトリ内の他のファイル名と重複しないファイル名を生成し，この領域に格納します．作成されたファイルはコンパチブル・モードでオープンされます．プログラム中で一時ファイルが必要な場合にこのシステム・コールを使用してファイル名の重複を避けることができます．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

5BH　新規ファイルの作成

CXレジスタに作成するファイルの属性，DS／DXにパス名を設定して行います．AXレジスタにファイル・ハンドルが返されます．作成されたファイルはコンパチブル・モードでオープンされます．同一のファイル名が存在した場合にはエラーとなります．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

5CH　ファイル・アクセスのロック／ロック解除

ALレジスタに00 Hを設定した場合にロック，01 Hの場合にロック解除となります．BXレジ

スタにファイル・ハンドル，CX／DXにロックまたはロック解除する領域のオフセット・アドレス，SI／DIにロックまたはロック解除する領域の長さを設定して行います．エラー発生時にはキャリー・フラグをセットし，AXレジスタにエラー・コードが返されます．

5EH　マシン名の取得／プリンタのセットアップ

ALレジスタに00Hを設定した場合，マシン名の取得になります．DS／DXに16バイトのバッファ領域のオフセット・アドレスを設定して行います．この領域にローカル・コンピュータのネット名が，CXレジスタにローカル・コンピュータの番号が返されます．このシステム・コールはMS-Networks稼働中にしか使用できません．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

ALレジスタに02Hを設定した場合，プリンタのセットアップになります．BXレジスタにプリンタの割り当てリストの中のインデックスを，CXレジスタにセットアップする文字列の長さを，DS／SIにセットアップする文字列のオフセット・アドレスを設定して行います．このシステム・コールはMS-Networks稼働中にしか使用できません．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

5FH　割り当てエントリの操作

ALレジスタに設定する値(コード)によって3種類の機能を持ちます．

AL：02H　割り当てリストのエントリの取得
AL：03H　割り当てリストのエントリの作成
AL：04H　割り当てリストのエントリのキャンセル

02H　ネットワークの割り当てリストのエントリを得る

BXレジスタに割り当てリストのインデックスを，DS／SIに16バイトのバッファのオフセット・アドレスを，ES／SIに128バイトのバッファのオフセット・アドレスを設定して行います．DS／SIに示される領域にローカル名(ソース・デバイス名)を，ES／SIに示される領域にリモート名(デスティネーション・デバイス名)が返されます．BLレジスタには，ローカルがプリンタの場合は03Hが，ディスク・ドライブの場合は04Hが返されます．CXレジスタにはコード03Hのシステム・コールで設定されたユーザー変数が返されます．このシステム・コールはMS-Networks稼働中にしか使用できません．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

03H　ソース・デバイス(プリンタまたはディスク・ドライブ)をネットワーク・デバイス(デスティネーション・デバイス)としてリダイレクトする

BLレジスタに，ソース・デバイスがプリンタの場合は03Hを，ディスク・ドライブの場合は04Hを設定します．CXレジスタにはユーザー変数を，DS／SIにソース・デバイス名のオフセット・アドレスを，ES／DIにデスティネーション・デバイス名のオフセット・アドレスを設定して行います．このシステム・コールはMS-Networks稼働中にしか使用できません．エラー

発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

04H　コード03Hで作成されたリダイレクトのキャンセル

DS／SIにキャンセルするプリンタ名またはディスク・ドライブ名を設定して行います．このシステム・コールはMS-Networks稼働中にしか使用できません．エラー発生時にはキャリーフラグをセットし，AXレジスタにエラー・コードが返されます．

62H　PSPの取得

AHレジスタ以外の入力パラメータは必要ありません．BXレジスタにカレント・プロセスのPSPのセグメント・アドレスが返されます．

ない場合はキャリーフラグをセットし、AXレジスタにエラーコードが返されます。

04H コード03Hで作成されたリダイレクトのキャンセル

DS：SIにキャンセルするデバイス名またはディスクドライブ名を設定しておきます。このファンクションはMS-Networks稼働中にしか使用できません。エラー発生時にはキャリーフラグをセットし、AXレジスタにエラーコードが返されます。

62H PSPの取得

AHレジスタ以外の入力パラメータは必要ありません。BXレジスタにカレントプロセスのPSPのセグメントアドレスが返されます。

拡張インターフェースボード周辺機器の利用

第4章

4-1　16色グラフィックボード対応タイニーグラフィックツール

4-1-1　16色グラフィックボード(PC-9801-24)の説明

16色グラフィックボード(グラフィックチャージャ)は，PC-9800シリーズのパソコン本体裏面の拡張用スロットに実装することによって，システムのグラフィック機能モードの拡張グラフィックモードにおいて，4096色中・16色モードを選択することが可能になります．このモードでは，4096色中の任意の16色を選んで表示することができます．

ただし，16色グラフィックボード(PC-9801-24)を使用して4096色中・16色モードを選択するには，アナログRGB対応ディスプレイが接続されていなければなりません．当然のことながら，PC-9800シリーズの従来機(PC-9801／E／F／M)では使用できません．

4-1-1-1　コマンド

16色グラフィックボードは内部にWrite Onlyのレジスタを2つ持っています．

命　　令	I／Oアドレス	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ライトモードレジスタ	7C	CGモード	RMWモード	0	0	$\overline{\text{P3EN}}$	$\overline{\text{P2EN}}$	$\overline{\text{P1EN}}$	$\overline{\text{P0EN}}$
ライトタイルレジスタ	7E	タイルレジスタ0～3(注)							

表4-1-1　GRCGのレジスタ

注：モードレジスタにライトを行うと，タイルレジスタ0ライトにリセットされます．その後ライトするたびに，書かれるレジスタは1,2，3,0と変化します．

ビット名	ビット＝1の意味	ビット＝0の意味
CGモード	GRCGを有効とする． CPUのVRAMアクセスをきっかけとして,GRCGが各モードの動作を実行する．	GRCGを無効とする． CPUのVRAMアクセスは，そのままVRAMのリード(ライト)となる．
RMWモード	CPUのVRAMライトにより，RMWモードの動作を行う． CPUのVRAMリードは無視される．	CPUのVRAMライトによりTDWモードの動作を行う． CPUのVRAMリードによりTCRモードの動作を行う．
P3EN，P2EN P1EN，P0EN	該当するプレーンを無効とする．	該当するプレーンを有効とする． 複数ビットの指定が可能． GRCGは，有効となっているプレーンに対してのみアクセスを行う．

表4-1-2　GRCGのモードレジスタ

・16色グラフィックボード使用中はCPUにWAITがかかり，モードレジスタの変更ができなくなります．

・16色グラフィックボード使用中はバスが占有されるため，DMA転送レートが低下するので注意が必要です．

4-1-1-2　モード

● TDWモード

VRAMに対してCPUが書き込みを行うと，CPUのライトデータは無視され，タイルレジスタの内容が各プレーンに書き込まれます．

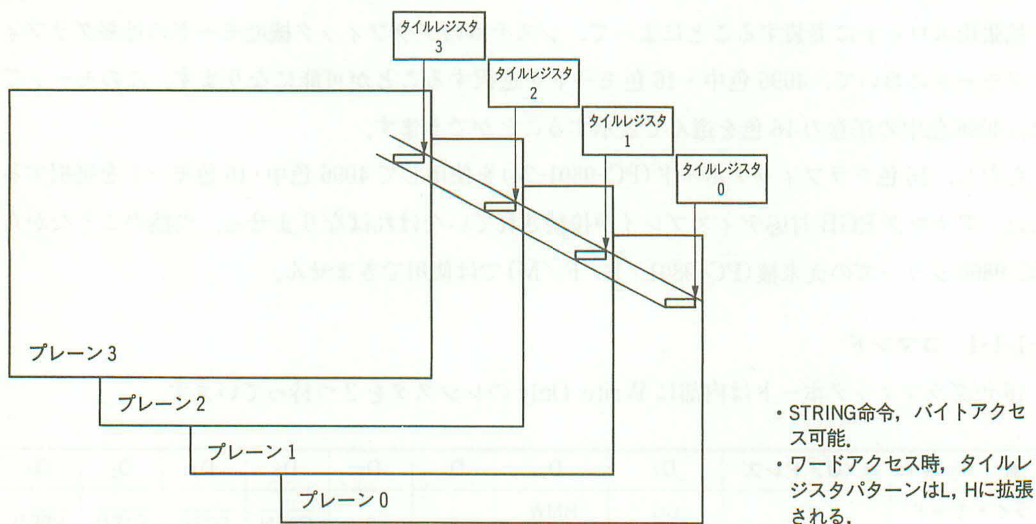


図4-1-1　TDWモード・プレーン

● TCRモード

CGモード設定後，該当アドレスをCPUが読み込むと，各プレーンとタイルレジスタの一致したビットを取り，すべてのプレーンで一致したビットを1にしてリードデータとしてCPUに返します．

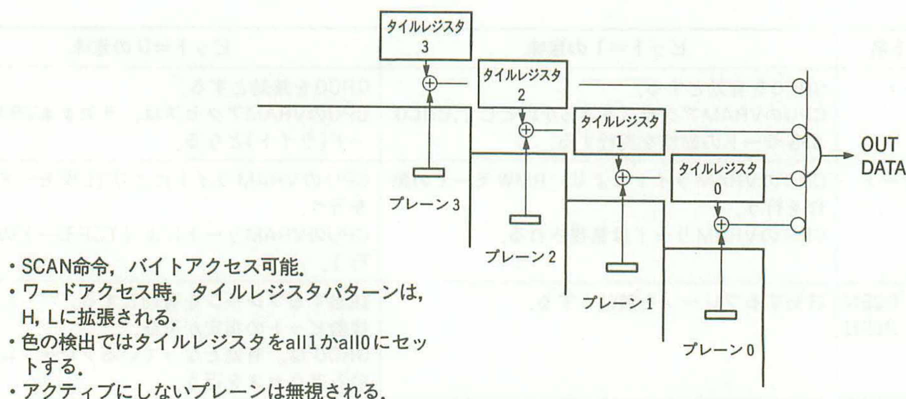


図4-1-2　TCRモード・プレーン

● RMW モード

RMW モード設定後，VRAM に対して CPU が書き込みを行うと，ライトデータの 1 のビットはタイルレジスタの内容が書き込まれ，0 のビットは元のデータが残されます．

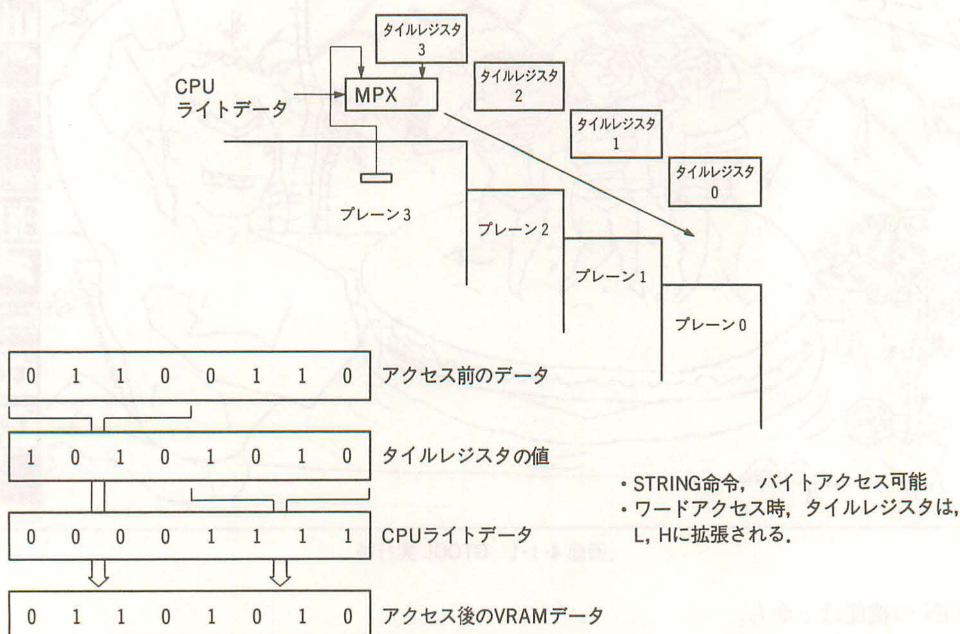


図 4-1-3　RMW モード・プレーン

4-1-2　応用プログラム

この項では，16 色グラフィックボードを使用した BASIC のサンプルプログラム，「Tiny GRAPHIC TOOL」を紹介します．

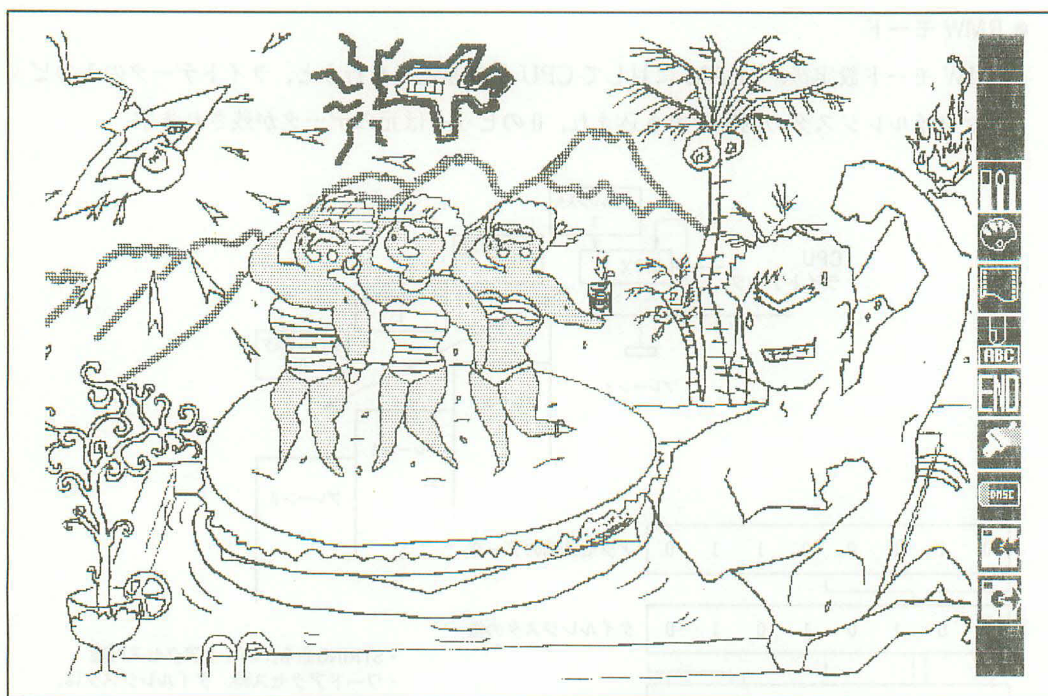
4-1-2-1　プログラムの説明

プログラムはマウスインターフェース部分，GTOOL 本体，アイコンのデータの 3 つの部分で構成されています．

ここに上げるプログラムリストは，MS-DOS 版 N 88-BASIC(86)のソースリストです．インタープリタでも動作しますが，MS-DOS 版 N 88-日本語 BASIC(86)コンパイラをお持ちの方は，コンパイルをかけた方が実行速度の面でより使い勝手が良くなると思われます．

4-1-2-2　使用法の説明

プログラムは，ほとんどの動作をマウスで選択／実行します．画面右端が実行を制御するアイコンと現在選択されている色です．



画面 4-1-1　GTOOL 実行時

ICON の機能は上から,

・現在選択している色，パレットカラーを変更する ICON
・現在選択しているペンの状態，太さを変更する ICON
・選択パレットを変更する ICON
・プリンタへハードコピーする ICON
・文字列を画面に書き込む ICON
・終了 ICON
・PAINT ICON
・画面初期化 ICON
・画面を DISK に SAVE する ICON
・SAVE された画面を LOAD する ICON

となっています．初期状態では，ペンによる描画ができる状態になっています．マウスの左ボタンを押すことによって画面に点が描かれ，ボタンを放さずにマウスを移動させると線が引けます．点または線の色を変更するには，カラー変更 ICON をマウスの左ボタン，または右ボタンでクリックします．それにより，色が変更されます．画面右上の現在の色を表す ICON で色を確認してください．

ペンは太さが8種類あり，ペンの太さを変更する ICON をクリックすることにより変更できます(左ボタンで太く，右ボタンで細くなります)．

4-1-2-3　ICON による動作の説明

色選択 ICON，ペン変更 ICON は上記を参考にしてください．

●パレット変更 ICON

現在選択している色を微調整し，好みの色を作ります．縦に3本並んでいるバーをマウスでクリックすることにより，色を混合して作り出します．

●文字列入力 ICON

この ICON をマウスでクリックすることにより，文字入力モードに入ります．漢字(NECDIC 漢字変換プログラムが必要)または ANK 文字を入力し，その後，マウスを動かし，文字列を表示したい位置でクリックすることにより，文字列を画面に表示します．

●終了 ICON

この ICON を選択するとプログラムは終了します．

● PAINT ICON

この ICON を選択したあと画面に戻り，塗りつぶしたい位置を指定すると黒(0)を境界色として囲まれた部分を現在の選択色で塗りつぶします．

●画面初期化(消しゴム)ICON

この ICON を2回選択すると画面が初期状態に戻ります．

●画面 SAVE ICON

画面を保存します．

●画面 LORD ICON

保存された画面を再生します．

プログラム 4-1-1　GTOOL.BAS

```
1000 ' save "gtool.bas",a
1010 '
1020 ' >>>======================================<<<
1030 ' >>>       graphics tool ver 1.0          <<<
1040 ' >>> ver 1.0 by BOGY. copyright (C) DMSC <<<
1050 ' >>>======================================<<<
1060 '
1070 '
1080 ' init
1090    DEFINT A-Z
1100    ON ERROR GOTO *ER
1110    COLOR.MAX=8                         ' 16色グラフィックボード無し。
1120 '  COLOR.MAX=16                        '                       有り。
1130    GOSUB *M.INIT                       ' マウス、画面等の初期化
1140    PTR=0
1150 '
1160    DIM PAT(500),P(3),PAL$(16)
1170    GOSUB *INIT.PAT                     ' マウスカーソルの設定
1180 '
1190    PEN.SIZE = 8
1200    PEN.COLOR = 0
1210    BACK.COLOR=7
```

```
1220    SAV.F=0
1230    STR.F=0
1240    PAT.F=0
1250 '
1260    GOSUB *PUT.ICON                         ' アイコンを表示する
1270    GOSUB *PUT.INF
1280 '
1290 *MAIN.LOOP
1300    IF PTR=M.MC THEN *MAIN.LOOP
1310    MOUSE 1,,,0
1320    IF M.X(PTR)<608 AND STR.F+PAT.F=0 GOTO 1390
1330    IF STR.F=1 THEN GOSUB *STR.SUB:GOTO 1420
1340    IF PAT.F=1 THEN GOSUB *PAT.SUB:GOTO 1420
1350    IF M.Y(PTR)>32 AND M.Y(PTR)<64 THEN GOSUB *PALL:GOTO 1420
1360    ON (ABS(M.Y(PTR))-80)¥32+1 GOSUB *PEN,*COL,*CPY,*STR,*QUIT,*PAINT.SUB,*CLR,
*SAV,*LOD:GOTO 1420
1370    IF STR.F+PAT.F=0 GOTO 1390
1380 '                                        ドットの表示
1390    IF M.X(PTR)>=75*8 THEN M.X(PTR)=75*8-1
1400    PUT(M.X(PTR),M.Y(PTR)),PAT,PSET,PEN.COLOR,BACK.COLOR
1410 '
1420    PTR=PTR+1
1430    IF PTR=M.TMAX THEN PTR=0
1440    IF PTR<>M.MC THEN 1320
1450    MOUSE 1,,,1
1460    SAV.F=0
1470    GOTO *MAIN.LOOP
1480 '
1490 *INIT.PAT                              ' ドットパターンの設定
1500    FOR I=0 TO  500
1510      PAT(I)=&HFFFF
1520    NEXT
1530    P(0)=0
1540    PAT(0)=8
1550    PAT(1)=8
1560 '
1570    RESTORE 1620
1580    FOR I=0 TO 15
1590     READ PAL$(I)
1600    NEXT
1610    RETURN                                 ' パレットの初期設定データ
1620    DATA "000","00F","0F0","0FF","F00","F0F","FF0","FFF"
1630    DATA "000","00A","0A0","0AA","A00","A0A","AA0","AAA"
1640 '
1650 *PEN' ペンの太さを設定する
1660    IF M.F(PTR)=1 THEN 1690
1670    IF PEN.SIZE=1 THEN PEN.SIZE=COLOR.MAX ELSE PEN.SIZE=PEN.SIZE-1
1680    GOTO 1700
1690    IF PEN.SIZE=COLOR.MAX THEN PEN.SIZE=1 ELSE PEN.SIZE=PEN.SIZE+1
1700    PAT(0)=PEN.SIZE : PAT(1)=PEN.SIZE
1710    KEY 1,"PEN"+STR$(PEN.SIZE)
1720    M.MVS=8+(8-PEN.SIZE)*3
1730    MOUSE 3,1,M.MVS
1740    MOUSE 3,0,M.MVS
1750    MOUSE 4,1,1,638-PEN.SIZE,398-PEN.SIZE
1760    GOSUB *PUT.INF
1770    RETURN
1780 '
1790 *COL                                   ' カラーパレット番号の変更
1800    IF M.F(PTR)=1 THEN 1840
1810    PEN.COLOR=PEN.COLOR-1
1820    IF PEN.COLOR=-1 THEN PEN.COLOR=COLOR.MAX-1
1830    GOTO 1860
1840    PEN.COLOR=PEN.COLOR+1
```

```
1850    IF PEN.COLOR=COLOR.MAX THEN PEN.COLOR=0
1860    KEY 2,"COL"+STR$(PEN.COLOR)
1870    KEY 3,"PAL"+PAL$(PEN.COLOR)
1880    GOSUB *PUT.INF
1890    RETURN
1900 '
1910 *PALL                                    ' パレット色の変更
1920    N=1 : PAL$=PAL$(PEN.COLOR)
1930 *PAL.LOOP
1940    K=(M.X(PTR)-76*8)/10+1
1950    PAL$="0123456789ABCDEF"
1960    K$=MID$(PAL$(PEN.COLOR),K,1)
1970    A=INSTR(PAL$,K$)
1980    IF M.F(PTR)=1 THEN 2010
1990    IF A=1 THEN A=16 ELSE A=A-1
2000    GOTO 2020
2010    IF A=16 THEN A=1 ELSE A=A+1
2020    MID$(PAL$(PEN.COLOR),K,1)=MID$(PAL$,A,1)
2030    KEY 3,"PAL"+PAL$(PEN.COLOR)
2040    COLOR=(PEN.COLOR,VAL("&h"+PAL$(PEN.COLOR)))
2050    RETURN
2060 '
2070 *CPY                                     ' プリンターへのハードコピー
2080    MOUSE 1,,,0
2090    COPY 2
2100    MOUSE 1,,,1
2110    RETURN
2120 '
2130 *STR                                     ' 文字列を画面に書き込む
2140    LOCATE 0,23
2150    CONSOLE ,,1
2160    LINE INPUT A$
2170    CONSOLE ,,0
2180    IF A$="" THEN RETURN
2190    STR.F=1:RETURN
2200 '
2210 *STR.SUB
2220    X=M.X(PTR):Y=M.Y(PTR)
2230    IF X>=76*8 THEN RETURN
2240    FOR I=1 TO KLEN(A$)
2250      B$=KMID$(A$,I,1)
2260      IF KTYPE(B$,1)=0 THEN B$=STR$(ASC(B$)) ELSE B$="&H"+JIS$(B$)
2270      IF X+16>=640 OR Y+16>=400 THEN 2300
2280      PUT(X,Y),KANJI(VAL(B$)),PSET,PEN.COLOR,BACK.COLOR
2290      IF KTYPE(KMID$(A$,I,1),1)=1 THEN X=X+16 ELSE X=X+8
2300    ' stop
2310    NEXT
2320    MOUSE 1,,,1
2330    CLS:STR.F=0
2340    RETURN
2350 '
2360 *SAV                                     ' グラフィックデータをディスクに格納
2370    IF SAV.F=1 THEN RETURN
2380    MOUSE 1,,,0
2390    DEF SEG=&HA800
2400    BSAVE "gdata.vr0",&H0,&H7FFF
2410    DEF SEG=&HB000
2420    BSAVE "gdata.vr1",&H0,&H7FFF
2430    DEF SEG=&HB800
2440    BSAVE "gdata.vr2",&H0,&H7FFF
2445    IF COLOR.MAX=8 THEN 2470
2450    DEF SEG=&HE000
2460    BSAVE "gdata.vr3",&H0,&H7FFF
2470    OPEN "gdata.pal" FOR OUTPUT AS #1
```

```
2480     FOR I=0 TO 15 : PRINT #1,PAL$(I) : NEXT
2490     CLOSE #1
2500     MOUSE 1,,,1
2510     PTR=M.MC-1:M.CR=0:M.CL=0
2520     RETURN
2530 '
2540 *LOD' グラフィックデータを取り込む
2550     MOUSE 1,,,0
2560     DEF SEG=&HA800
2570     BLOAD "gdata.vr0",&H0
2580     DEF SEG=&HB000
2590     BLOAD "gdata.vr1",&H0
2600     DEF SEG=&HB800
2610     BLOAD "gdata.vr2",&H0
2615     IF COLOR.MAX=8 THEN 2640
2620     DEF SEG=&HE000
2630     BLOAD "gdata.vr3",&H0
2640     OPEN "gdata.pal" FOR INPUT  AS #1
2650     FOR I=0 TO COLOR.MAX-1
2652        INPUT #1,PAL$(I)
2654        COLOR=(I,VAL("&H"+PAL$(I)))
2656     NEXT
2660     CLOSE #1
2670     MOUSE 1,,,1
2680     GOSUB *PUT.INF
2690     PTR=M.MC-1:M.CR=0:M.CL=0
2700     RETURN
2710 '
2720 *QUIT' 終了
2730     MOUSE 1,,,0
2740     CLS 3
2750     END
2760 '
2770 *PAINT.SUB' 画面を塗り潰す、境界色は黒（0）
2780     PAT.F=1
2790     RETURN
2800 '
2810 *PAT.SUB
2820     MOUSE 1,,,0
2830     IF MOUSE(0)>=76*8 THEN RETURN
2840     MOUSE 1,,,0
2850     PAINT(MOUSE(0),MOUSE(1)),PEN.COLOR,0
2860     MOUSE 1,,,1:PAT.F=0
2870     RETURN
2880 '
2890 *CLR' 画面を初期状態に戻す。
2900     CLRF=CLRF+1:IF CLRF=2 THEN CLRF=0 ELSE RETURN
2910     MOUSE 1,,,0
2920     LINE(0,0)-(75*8+6,399),7,BF
2930     MOUSE 1,,,1
2940     RETURN
2950 '
2960 '                              マウスインターフェース
2970 *M.INIT
2980     GOSUB *INIT.SCREEN
2990     GOSUB *INIT.VAR
3000     GOSUB *INIT.MOUSE
3010     RETURN
3020 '
3030 '
3040 *INIT.SCREEN' グラフィックの初期設定
3050     CONSOLE 0,25,0,1
3060     SCREEN 3,0
3070     LINE(0,0)-(639,399),7,BF
```

```
3080    MOUSE 0
3090    IF COLOR.MAX=8 THEN COLOR 1,,,,1 ELSE COLOR 1,,,,2
3100    CLS
3110    RETURN
3120 '
3130 '
3140 *INIT.VAR' マウス関係の変数の設定
3150    DIM M.Y(500),M.X(500),M.FL(500),M.F(500),M.FONT$(64)
3160    M.TMAX=500                  ' trace ﾍﾝｽｳ ﾉ max
3170    M.MC=0                      ' mouse ｶﾞ ｳｺﾞｲﾀ ｷｾｷ ﾍﾉ ﾎﾝﾀ
3180    M.CL=0                      ' mouse left ﾎﾞﾀﾝ ﾉ ｸﾘｯｸ ｼﾞｮｳﾀｲ
3190    M.CR=0                      '           right
3200    M.MV=0
3210    RETURN
3220 '
3230 '
3240 *INIT.MOUSE' マウスの初期設定。
3250    MOUSE 0
3260    GOSUB *M.SET.FONT
3270    MOUSE 4,0,0,639,399
3280    ON MOUSE(1) GOSUB *M.MOVE        : MOUSE(1) ON
3290    ON MOUSE(2) GOSUB *M.CLICK.LEFT  : MOUSE(2) ON
3300    ON MOUSE(3) GOSUB *M.CLICK.RIGHT : MOUSE(3) ON
3310    ON MOUSE(4) GOSUB *M.CLOFF.LEFT  : MOUSE(4) ON
3320    ON MOUSE(5) GOSUB *M.CLOFF.RIGHT : MOUSE(5) ON
3330    MOUSE 1,,,1
3340    RETURN
3350 '
3360 '
3370 *M.MOVE                                ' マウス移動処理
3380    IF M.CL+M.CR=0 OR MOUSE(0)>607 THEN RETURN
3385 *M.MOVE.1
3390    M.X(M.MC)=MOUSE(0)
3400    M.Y(M.MC)=MOUSE(1)
3410    M.F(M.MC)=M.CL+M.CR*2
3420    M.MC=M.MC+1 : IF M.MC=M.TMAX THEN M.MC=0
3430    RETURN
3440 '
3450 '                                      マウスクリック処理
3460 *M.CLICK.LEFT
3470    M.CL=1:GOTO *M.MOVE.1
3480 '
3490 '
3500 *M.CLICK.RIGHT
3510    M.CR=1:GOTO *M.MOVE.1
3520 '
3530 '
3540 *M.CLOFF.LEFT
3550    M.CL=0:RETURN
3560 '
3570 '
3580 *M.CLOFF.RIGHT
3590    M.CR=0:RETURN
3600 '
3610 *M.SET.FONT' マウスカーソルデータ
3620 DATA FC,00,F8,00,F0,00,F8,00,DC,00,8E,00,07,00
3630 DATA 02,00,00,00,00,00,00,00,00,00,00,00,00,00
3640 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
3650 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
3660 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
3670 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
3680 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
3690 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

```
3700 '
3710    RESTORE 3620
3720    M.FONT$=""
3730    FOR I=0 TO 63
3740      READ A$
3750      M.FONT$=M.FONT$+CHR$(VAL("&H"+A$))
3760    NEXT
3770    MOUSE 2,0,0,M.FONT$
3780    RETURN
3790 '
3800 *PUT.ICON:' ICON DISPLAY
3810    RESTORE *ICON.DAT:DIM T%(300)
3820    Y=5*16:X=38*16
3830    FOR K=0 TO 8
3840         READ T%(0),T%(1)
3850         FOR L=0 TO 8*24-1
3860              READ A$
3870              T%(2+L)=VAL("&H"+A$)
3880         NEXT
3890         PUT(X,Y),T%,PSET
3900         Y=Y+32
3910    NEXT
3920    RETURN
3930 '
3940 *PUT.INF' 情報表示ルーチン
3950    LINE(38*16,0)-STEP(31,31),0,BF
3960    LINE(38*16,0)-STEP(31,31),7,B
3970    PAINT(38*16+1,1),PEN.COLOR,7
3980    LINE(38*16,0)-STEP(31,31),0,B
3990 '
4000    LINE(38*16,32)-STEP(10,32),4,BF
4010    LINE STEP(1,0)-STEP(10,-32),2,BF
4020    LINE STEP(1,0)-STEP(10,32),1,BF
4030 '
4040    LINE(77*8+4,65)-STEP(10,10),7,BF
4050    LINE(76*8-1,0)-STEP(0,399),0
4060    PUT(77*8+8-PAT(0)/2,64+8-PAT(1)/2),PAT,PSET,PEN.COLOR,BACK.COLOR
4070    RETURN
4080 '
4090 *ER
4095    BEEP
4100    PTR=M.MC
4110    MOUSE 1,,,1
4120    RESUME *MAIN.LOOP
4130 '
4140 *ICON.DAT' アイコンのグラフィックデータ
4150 DATA 32, 32
4160 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
4170 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4180 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4190 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4200 DATA FF7F,FCFF,FF7F,FCFF,3E60,FC7F,3E60,FC7F
4210 DATA 3E60,FC7F,BC6F,FC3F,BC6F,FC3F,BC6F,FC3F
4220 DATA B96F,FC9F,B96F,FC9F,B96F,FC9F,BB6F,3CDC
4230 DATA BB6F,3CDC,BB6F,3CDC,BB6F,3CDC,BB6F,3CDC
4240 DATA BB6F,3CDC,BB6F,3CDC,BB6F,3CDC,BB6F,3CDC
4250 DATA BB6F,3CDC,BB6F,3CDC,BB6F,3CDC,BB6F,3CDC
4260 DATA BB6F,3CDC,BB6F,3CDC,B96F,3C9C,B96F,3C9C
4270 DATA B96F,3C9C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4280 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4290 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4300 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4310 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4320 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
```

```
4330 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4340 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4350 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4360 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4370 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4380 DATA 3C60,3C3C,3C60,3C3C,3C60,3C3C,3C60,3C3C
4390 DATA 3C60,3C3C,0000,0000,0000,0000,0000,0000
4400 '     ICON2.DATA
4410 DATA 32, 32
4420 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
4430 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4440 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4450 DATA F07F,FC3F,F07F,FC3F,F07F,FC3F,8F7F,FCC7
4460 DATA 8F7F,FCC7,8F7F,FCC7,7E7F,FC7B,7E7F,FC7B
4470 DATA 7E7F,FC7B,847C,FC24,847C,FC24,847C,FC24
4480 DATA 867B,7C63,867B,7C63,867B,7C63,827B,7C41
4490 DATA 827B,7C41,827B,7C41,C375,BC43,C375,BC43
4500 DATA C375,BC43,E370,BCC7,E370,BCC7,E370,BCC7
4510 DATA 7B68,DCCF,7B68,DCCF,7B68,DCCF,3F68,DCFF
4520 DATA 3F68,DCFF,3F68,DCFF,3F6E,DCFF,3F6E,DCFF
4530 DATA 3F6E,DCFF,FF6F,DCFF,FF6F,DCFF,FF6F,DCFF
4540 DATA FF6F,DCFF,FF6F,DCFF,FF6F,DCFF,FF77,3CFE
4550 DATA FF77,3CFE,FF77,3CFE,FE77,FC7D,FE77,FC7D
4560 DATA FE77,FC7D,FD7B,FCB3,FD7B,FCB3,FD7B,FCB3
4570 DATA FB7B,FCAF,FB7B,FCAF,FB7B,FCAF,FD7C,FCAF
4580 DATA FD7C,FCAF,FD7C,FCAF,7E7F,FC5F,7E7F,FC5F
4590 DATA 7E7F,FC5F,8F7F,FCDF,8F7F,FCDF,8F7F,FCDF
4600 DATA F07F,FC3F,F07F,FC3F,F07F,FC3F,FF7F,FCFF
4610 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4620 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4630 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4640 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4650 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
4660 ' ICON4 DATA
4670 DATA 32, 32
4680 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
4690 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4700 DATA FF7F,FCFF,0060,0C00,0060,0C00,0060,0C00
4710 DATA FF6F,ECFF,FF6F,ECFF,FF6F,ECFF,FF69,2CFF
4720 DATA FF69,2CFF,FF69,2CFF,FF69,2CFF,FF69,2CFF
4730 DATA FF69,2CFF,FF6F,ECFF,FF6F,ECFF,FF6F,ECFF
4740 DATA FF69,2CFF,FF69,2CFF,FF69,2CFF,FF69,2CFF
4750 DATA FF69,2CFF,FF69,2CFF,FF6F,ECFF,FF6F,ECFF
4760 DATA FF6F,ECFF,FF69,2CFF,FF69,2CFF,FF69,2CFF
4770 DATA FF69,2CFF,FF69,2CFF,FF69,2CFF,FF6F,ECFF
4780 DATA FF6F,ECFF,FF6F,ECFF,FF69,2CFF,FF69,2CFF
4790 DATA FF69,2CFF,FF69,2CFF,FF69,2CFF,FF69,2CFF
4800 DATA FF6F,ECFF,FF6F,ECFF,FF6F,ECFF,FF69,2CFF
4810 DATA FF69,2CFF,FF69,2CFF,FF69,2CFF,FF69,2CFF
4820 DATA FF69,2CFF,0768,ECFF,0768,ECFF,0768,ECFF
4830 DATA F963,2CFF,F963,2CFF,F963,2CFF,FE6F,2CFF
4840 DATA FE6F,2CFF,FE6F,2CFF,FF7F,9C3F,FF7F,9C3F
4850 DATA FF7F,9C3F,FF7F,7CC0,FF7F,7CC0,FF7F,7CC0
4860 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4870 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4880 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4890 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4900 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
4910 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
4920 ' ICON 5 DATA
4930 DATA 32, 32
4940 DATA 0000,0000,0000,0000,0000,0000,BF7F,FCEF
4950 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,BF7F,FCEF
4960 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,BF7F,FCEF
```

```
4970 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,BF7F,FCEF
4980 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,BF7F,FCEF
4990 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,BF7F,FCEF
5000 DATA BF7F,FCEF,BF7F,FCEF,BF7F,FCEF,837F,FCEF
5010 DATA 837F,FCEF,837F,FCEF,BB7F,FCEF,BB7F,FCEF
5020 DATA BB7F,FCEF,BB7F,FCEF,BB7F,FCEF,BB7F,FCEF
5030 DATA BB7F,FCEF,BB7F,FCEF,BB7F,FCEF,BB7F,FCEF
5040 DATA BB7F,FCEF,BB7F,FCEF,BB7F,FCEF,BB7F,FCEF
5050 DATA BB7F,FCEF,BB7F,FCEF,BB7F,FCEF,BB7F,FCEF
5060 DATA D77F,FCDF,D77F,FCDF,D77F,FCDF,EF7F,FCBF
5070 DATA EF7F,FCBF,EF7F,FCBF,F07F,FC7F,F07F,FC7F
5080 DATA F07F,FC7F,0040,0400,0040,0400,0040,0400
5090 DATA FF5F,F4FF,FF5F,F4FF,FF5F,F4FF,185E,F470
5100 DATA 185E,F470,185E,F470,C95C,7426,C95C,7426
5110 DATA C95C,7426,C95C,7426,C95C,7426,C95C,7426
5120 DATA C85C,F467,C85C,F467,C85C,F467,095C,F427
5130 DATA 095C,F427,095C,F427,C95C,7426,C95C,7426
5140 DATA C95C,7426,C95C,7426,C95C,7426,C95C,7426
5150 DATA C85C,F470,C85C,F470,C85C,F470,FF5F,F4FF
5160 DATA FF5F,F4FF,FF5F,F4FF,FF5F,F4FF,FF5F,F4FF
5170 DATA FF5F,F4FF,0000,0000,0000,0000,0000,0000
5180 ' ICON6 DATA
5190 DATA 32, 32
5200 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
5210 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5220 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5230 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,1340,1C90
5240 DATA 1340,1C90,1340,1C90,1340,0C90,1340,0C90
5250 DATA 1340,0C90,F34F,8C93,F34F,8C93,F34F,8C93
5260 DATA F14F,CC93,F14F,CC93,F14F,CC93,F14F,CC93
5270 DATA F14F,CC93,F14F,CC93,F04F,CC93,F04F,CC93
5280 DATA F04F,CC93,F04F,CC93,F04F,CC93,F04F,CC93
5290 DATA F04F,CC93,F04F,CC93,F04F,CC93,F24F,CC93
5300 DATA F24F,CC93,F24F,CC93,F24F,CC93,F24F,CC93
5310 DATA F24F,CC93,1240,CC93,1240,CC93,1240,CC93
5320 DATA 1240,CC93,1240,CC93,1240,CC93,F24F,CC93
5330 DATA F24F,CC93,F24F,CC93,F34F,CC13,F34F,CC13
5340 DATA F34F,CC13,F34F,CC13,F34F,CC13,F34F,CC13
5350 DATA F34F,CC13,F34F,CC13,F34F,CC13,F34F,CC13
5360 DATA F34F,CC13,F34F,CC13,F34F,CC13,F34F,CC13
5370 DATA F34F,CC13,F34F,CC13,F34F,CC13,F34F,CC13
5380 DATA F34F,CC13,F34F,CC13,F34F,CC13,F34F,8C93
5390 DATA F34F,8C93,F34F,8C93,1340,0C90,1340,0C90
5400 DATA 1340,0C90,1340,1C90,1340,1C90,1340,1C90
5410 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5420 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5430 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
5440 ' ICON 7 DATA
5450 DATA 32, 32
5460 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
5470 DATA FF7F,FCFF,FF7F,FCFF,FF7F,3CFF,FF7F,3CFF
5480 DATA FF7F,3CFF,FF7F,1CFE,FF7F,1CFE,FF7F,1CFE
5490 DATA FF7F,1CFE,FF7F,1CFE,FF7F,1CFE,FF7F,3CFC
5500 DATA FF7F,3CFC,FF7F,3CFC,F17F,A8A8,F17F,A8A8
5510 DATA F17F,A8A8,C07F,5455,C07F,5455,C07F,5455
5520 DATA 807F,A8AA,807F,A8AA,807F,A8AA,207F,5455
5530 DATA 207F,5455,207F,5455,307E,A80A,307E,A80A
5540 DATA 307E,A80A,187E,5415,187E,5415,187E,5415
5550 DATA 0C7F,A802,0C7F,A802,0C7F,A802,867F,5401
5560 DATA 867F,5401,867F,5401,837F,A800,837F,A800
5570 DATA 837F,A800,817F,5401,817F,5401,817F,5401
5580 DATA 007F,A880,007F,A880,007F,A880,007E,FCCF
5590 DATA 007E,FCCF,007E,FCCF,007C,FC1F,007C,FC1F
5600 DATA 007C,FC1F,0078,FC3F,0078,FC3F,0078,FC3F
```

```
5610 DATA 0778,FC7F,0778,FC7F,0778,FC7F,0F79,FCFF
5620 DATA 0F79,FCFF,0F79,FCFF,1F7C,FCFF,1F7C,FCFF
5630 DATA 1F7C,FCFF,3F7E,FCFF,3F7E,FCFF,3F7E,FCFF
5640 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5650 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5660 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5670 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5680 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5690 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
5700 ' ICON 8 DATA
5710 DATA 32, 32
5720 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
5730 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5740 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5750 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5760 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5770 DATA FF7F,FCFF,0070,0C00,0070,0C00,0070,0C00
5780 DATA 7F65,F4FF,7F65,F4FF,7F65,F4FF,FF4A,F4FF
5790 DATA FF4A,F4FF,FF4A,F4FF,7F55,F4FF,7F55,F4FF
5800 DATA 7F55,F4FF,FF4A,F4FF,FF4A,F4FF,FF4A,F4FF
5810 DATA 4D55,7444,4D55,7444,4D55,7444,D44A,F45D
5820 DATA D44A,F45D,D44A,F45D,5555,F445,5555,F445
5830 DATA 5555,F445,D54A,F475,D54A,F475,D54A,F475
5840 DATA 4D55,7444,4D55,7444,4D55,7444,FF4A,F4FF
5850 DATA FF4A,F4FF,FF4A,F4FF,7F55,F4FF,7F55,F4FF
5860 DATA 7F55,F4FF,FF4A,F4FF,FF4A,F4FF,FF4A,F4FF
5870 DATA 7F65,F4FF,7F65,F4FF,7F65,F4FF,0070,0C00
5880 DATA 0070,0C00,0070,0C00,FF7F,FCFF,FF7F,FCFF
5890 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5900 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5910 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5920 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5930 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5940 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
5950 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
5960 ' ICON9 DATA
5970 DATA 32, 32
5980 DATA 0000,0000,0000,0000,0000,0000,FF7F,FCFF
5990 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6000 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6010 DATA 0070,1C00,0070,1C00,0070,1C00,E077,1C00
6020 DATA E077,1C00,E077,1C00,E077,1C00,E077,1C00
6030 DATA E077,1C00,E077,1C00,E077,1C00,E077,1C00
6040 DATA E077,1800,E077,1800,E077,1800,0070,1000
6050 DATA 0070,1000,0070,1000,0770,0884,0770,0884
6060 DATA 0770,0884,0F70,38CC,0F70,38CC,0F70,38CC
6070 DATA 1F70,78EC,1F70,78EC,1F70,78EC,3F70,F8DF
6080 DATA 3F70,F8DF,3F70,F8DF,3F70,F83F,3F70,F83F
6090 DATA 3F70,F83F,3F70,F8FF,3F70,F8FF,3F70,F8FF
6100 DATA 3F70,F83F,3F70,F83F,3F70,F83F,3F70,F8DF
6110 DATA 3F70,F8DF,3F70,F8DF,1F70,78EC,1F70,78EC
6120 DATA 1F70,78EC,0F70,38CC,0F70,38CC,0F70,38CC
6130 DATA 0770,0884,0770,0884,0770,0884,0070,1000
6140 DATA 0070,1000,0070,1000,0170,1800,0170,1800
6150 DATA 0170,1800,0170,1C00,0170,1C00,0170,1C00
6160 DATA 0170,1C00,0170,1C00,0170,1C00,0170,1C00
6170 DATA 0170,1C00,0170,1C00,0170,1C00,0170,1C00
6180 DATA 0170,1C00,0070,1C00,0070,1C00,0070,1C00
6190 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6200 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6210 DATA FF7F,FCFF,0000,0000,0000,0000,0000,0000
6220 ' ICON 10 DATA
6230 DATA 32,32,0000,0000,0000,0000
6240 DATA 0000,0000,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
```

```
6250 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6260 DATA FF7F,FCFF,FF7F,FCFF,0070,1C00,0070,1C00
6270 DATA 0070,1C00,E077,1C00,E077,1C00,E077,1C00
6280 DATA E077,1C00,E077,1C00,E077,1C00,E077,1C00
6290 DATA E077,1C00,E077,1C00,E077,9C00,E077,9C00
6300 DATA E077,9C00,0070,DC00,0070,DC00,0070,DC00
6310 DATA 0770,EC80,0770,EC80,0770,EC80,1F70,E4C0
6320 DATA 1F70,E4C0,1F70,E4C0,1F70,F4E0,1F70,F4E0
6330 DATA 1F70,F4E0,3F70,F0C1,3F70,F0C1,3F70,F0C1
6340 DATA 3E70,F81F,3E70,F81F,3E70,F81F,3E70,FCFF
6350 DATA 3E70,FCFF,3E70,FCFF,3E70,F81F,3E70,F81F
6360 DATA 3E70,F81F,3F70,F0C1,3F70,F0C1,3F70,F0C1
6370 DATA 1F70,F4E0,1F70,F4E0,1F70,F4E0,0F70,E4C0
6380 DATA 0F70,E4C0,0F70,E4C0,0770,EC80,0770,EC80
6390 DATA 0770,EC80,0070,DC00,0070,DC00,0070,DC00
6400 DATA 0170,9C00,0170,9C00,0170,9C00,0170,1C00
6410 DATA 0170,1C00,0170,1C00,0170,1C00,0170,1C00
6420 DATA 0170,1C00,0170,1C00,0170,1C00,0170,1C00
6430 DATA 0170,1C00,0170,1C00,0170,1C00,0070,1C00
6440 DATA 0070,1C00,0070,1C00,FF7F,FCFF,FF7F,FCFF
6450 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,FF7F,FCFF
6460 DATA FF7F,FCFF,FF7F,FCFF,FF7F,FCFF,0000,0000
6470 DATA 0000,0000,0000,0000
```

4-2　スーパーインポーズボードの応用例

4-2-1　スーパーインポーズボードの説明

スーパーインポーズボード(PC-9801-25)は，PC-9800シリーズの画面出力と外部のビデオ映像を合成し，ビデオ信号として出力する機能を持っています．外部の信号はNTSC方式の標準ビデオ信号であれば，テレビ放送，ビデオ・カメラ，ビデオ・ディスク，VTR等から信号を入力することができます．

外部のビデオ入力がない場合でも，NTSCビデオ信号に変換されたパソコンの画像が出力されますので，パソコンの画像のみをVTRに録画することも可能です．

ただし，スーパーインポーズボード(PC-9801-25)は，PC-9800シリーズの従来機(PC-9801／E／F／M)では使用できませんので注意が必要です．また，スーパーインポーズするパソコン側の画面は640×200ドットの8色モードしか使えません．

スーパーインポーズボードを使用するときは，パソコン本体のディップスイッチSW1の1をOFFに，2をONにセットする必要があります．

スーパーインポーズボードには，表4-2-1，2，4，5に示すモード・レジスタがあり，OUT文によって切り換えることができます．

ポートアドレス	内　容							
ED	モード設定							
F5	D7	D6	D5	D4	D3	D2	D1	D0
スイッチによる切換	0	0	0	0	YM	CM	M1	M0

表4-2-1　モード・レジスタ①

合成映像の出力に関して，表4-2-2に示す3つのモードが選択できます．

M1, M0	モード	動作内容
1, 1	禁止	
1, 0	PC	パソコンの画像のみを表示
0, 1	IMPOSE	VIDEO入力された画像とパソコンの画像を合成表示
0, 0	VIDEO	VIDEO入力された画像のみを表示

表4-2-2　モード・レジスタ②

外部映像入力があり，リセット・ボタンが押された場合，自動的にIMPOSEモードになります．IMPOSEモードの判定は表4-2-3に示す端子で行います．

IMPOSEモードの判定	クロック入力端子
DOT CLOCK端子が常に1	DOT CLOCK

表4-2-3　IMPOSEモードの判定

VIDEO OUT出力に関して，表4-2-4に示すモードが選択できます．

C/M	モード	動作内容
1	COLOR	VIDEO OUT出力のパソコンの画像をカラーで表示
0	MONO	VIDEO OUT出力のパソコンの画像をモノクロで表示

表4-2-4　モード・レジスタ③

このモード選択によって，RGB MULTI端子，RGB OUT端子の画像は変化しません．また，横方向640ドットの中の1ドットづつに色を付けることはできません．

RGB MULTI出力に関して，表4-2-5に示すモードが選択できます．

YM	モード	動作内容
1	HARF BRIGHT	背景のビデオ映像を中間輝度で表示
0	NORMAL	背景のビデオ映像を普通輝度で表示

表4-2-5　モード・レジスタ④

4-2-2　応用例

4-2-2-1　接続方法

●外部ビデオ・ソースとの接続(パソコンに入力)

外部ビデオ・ソース(VTR，ビデオ・カメラ，ビデオ・ディスク，TVチューナー)との接続は，スーパーインポーズボードに付属しているフォノピンケーブルで，外部ビデオ・ソースの出力端子(カラーVBS，1 V_{P-P}，75 Ω)と，スーパーインポーズボードのVIDEO IN端子を接続します．スーパーインポーズ機能を使用せず，パソコンの出力をNTSC信号に変換するだけの場合は，この接続は必要ありません．外部ビデオ・ソースの音声出力は，モニタ・テレビの接続方法によって異なります．

● VIDEO OUT出力

スーパーインポーズボードに付属しているフォノピンケーブルで，モニタ・テレビ，VTRの入力端子と，スーパーインポーズボードのVIDEO OUT端子を接続します．VIDEO OUT端子には，NTSC信号が出力され，ビデオ録画したり，モニタ・テレビで入力されたビデオ映像を見ることができます．ただし，タイリング・ペイントによる中間色表現をした場合，色が乱れることがあります．

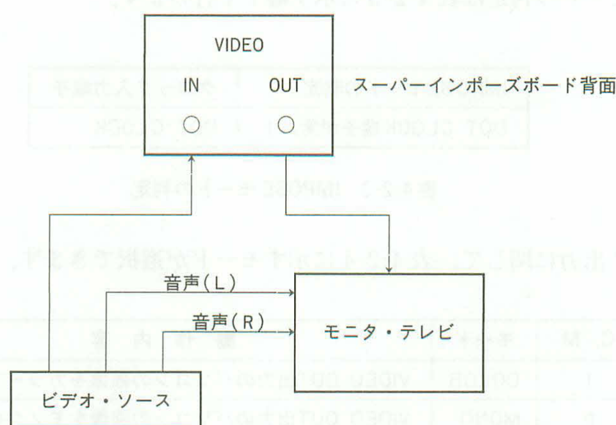


図4-2-1　VIDEO OUT接続

● 21ピンRGBマルチ出力

使用するモニタ・テレビが21ピンRGB対応の場合は，21ピンRGBマルチケーブル(市販)で，モニタ・テレビの21ピンRGB端子と，スーパーインポーズボードのRGB MULTI出力端子を接続し，モニタ・テレビの入力モードをRGBマルチにセットします．このモードでは外部入力された映像の上にRGBレベルでパソコンの画像が合成されるので，鮮明な合成画像を得ることができます．タイリング・ペイントによる中間色表現も自由に行うことができます．ただし，この合成画像をビデオ録画することはできませんので，注意が必要です．また，モニタ・テレビはRGBマルチコネクタ以外の入力を受け付けません．音声もマルチコネクタ経由の入力となるので，AUDIO IN端子に接続します．

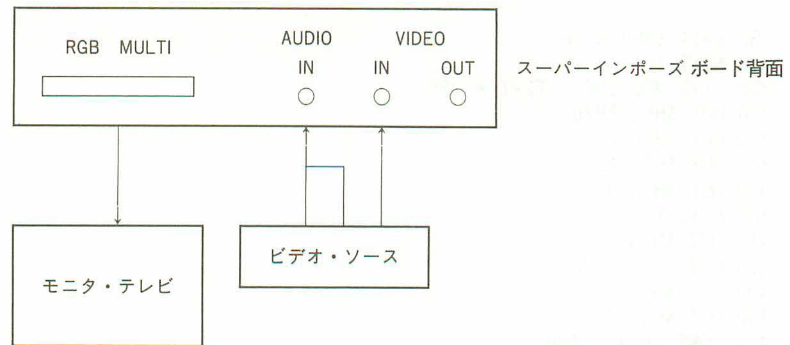


図 4-2-2　マルチコネクタ接続

● RGB OUT 出力

スーパーインポーズボードのRGB出力端子には，パソコンのRGB信号がそのまま出力されます．パソコンの専用ディスプレイを接続することにより，パソコンの画像を常時モニタすることができます．ただし，専用ディスプレイが高解像度ディスプレイの場合，スーパーインポーズモード時には同期がとれなくなりますので注意が必要です．

4-2-2-2　起動方法

接続及びパソコン本体の設定確認後，パソコン本体の電源をONにすると，通常の動作で立ち上がります．この時，外部映像機器が接続してあれば，すでにスーパーインポーズされた状態で立ち上がります．

プログラム 4-2-1　サンプルプログラム(smoth.bas)

```
100 ' smoth.bas ****************************************************************
110 '           ＢＡＳＩＣによるスムーススクロールプログラム
120 '               update 1986.09.23 copyright (C) DMSC
130 '
140 ' N88-ＢＡＳＩＣ（８６）コンパイラ（ＭＳ－ＤＯＳ版）でコンパイルできません
150 '*******************************************************************************
160 '
170 SCREEN 3,0,0,1 : WIDTH 80,25 : CONSOLE 0,25,0,1 : COLOR ,0
180 SCREEN ,,1 : CLS 3
190 SCREEN ,,0 : CLS 3
200 GOSUB *SCINIT
210 GOSUB *FLPRINT
220 GOSUB *SCROLL
230 END
240 '
250 *SCINIT
260 DEF SEG=&H60 : CLS 1 : POKE &H1412,&HFA : CLS 1 : RETURN
270 '
280 *SCROLL
290 OUT &H78,0 : OUT &H7A,24
300 FOR T=0 TO 25*80 STEP 80
310 FOR I=1 TO 15
320 OUT &H76,I
```

```
330 WAIT &H60,&H20
340 NEXT I
350 T1=T MOD 256 : T2=T ¥ 256
360 OUT &H62,&H70
370 OUT &H60,T1
380 OUT &H60,T2
390 OUT &H76,0
400 NEXT T
410 OUT &H76,0
420 OUT &H62,&H70
430 OUT &H60,0
440 OUT &H60,0
450 POKE &H1412,&H0
460 CLS
470 RETURN
480 '
490 *FLPRINT
500 OPEN "SMOTH.TXT" FOR INPUT AS #1
510 IF EOF(1) THEN 550
520 LINE INPUT #1,MESSAGE$
530 LOCATE 0,25: PRINT MESSAGE$
540 GOTO 510
550 CLOSE #1
560 RETURN
```

※Smoth. txtというファイル名で、スクロールさせたい文書を作成してください

4-3　サウンドボード

4-3-1　サウンドボードのハードウェア

別売のサウンドボードをPC 9801シリーズの本体に実装することによって，FM音源によ音楽演奏とMSX仕様のジョイスティクを入力装置として扱うことができます．このインターフェースボードは，音源チップ(YM-2203)と，コントロールプログラムを内蔵するROM(16 Kバイト)が実装されていて，そのブロックは以下の図のようになっています．

命　令	I/Oポート アドレス	R/W	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備　考
ライトアドレス	188	W	A7	A6	A5	A4	A3	A2	A1	A0	
ライトデータ	18A	W	D7	D6	D5	D4	D3	D2	D1	D0	
リードステータス	188	R	BUSY	×	×	×	×	×	FLAGB	FLAGA	
リードデータ	18A	R	D7	D6	D5	D4	D3	D2	D1	D0	ADDRESSは00～0Fに限る

表4-3-1　機能一覧

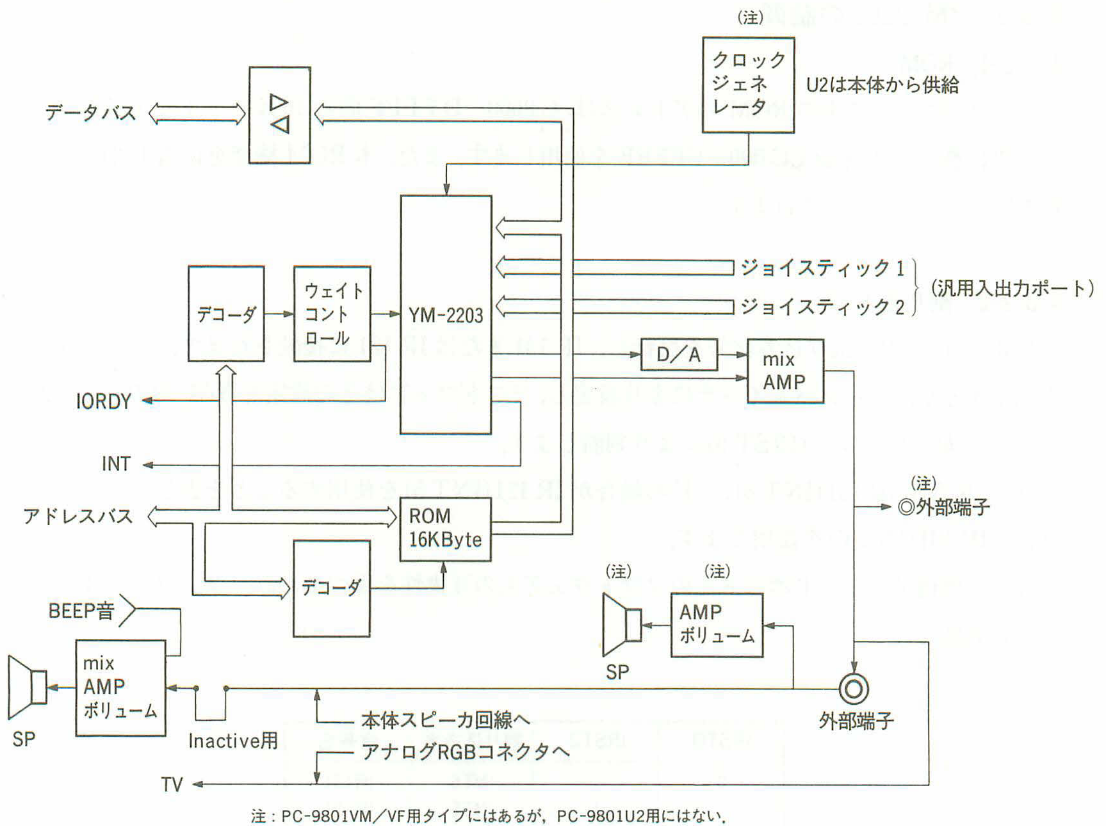


図 4-3-1　ブロック・ダイアグラム

注意：

1. 以上のコマンドを実行すると，7 または 8 クロック(従来の 2 クロックも含みます)ハードウェアで自動的にウェイトが入ります．
2. ADDRESS　WRITE でデータが 20 H～B 2 H の場合，次の DATA　WRITE まで 43 クロック(CPU クロック)以上のソフトウェアによるウェイトが必要です．
3. 指定されたアドレスが 20 H から B 2 H で DATA WRITE を実行した場合，次の 20 H～B 2 H に対する DATA　WRITE まで 208 クロック以上のソフトウェアによるウェイトが必要です．ただし，STATUS READ により BUSY フラグが 0 になったことを確認したのなら，ウェイトは必要ありません．
4. 2，3 のようなウェイトは，分周波数が 6 の場合であり，この値が〝3″になれば，2)では 23 クロック，3)では 105 クロックとなります．

   ＊ YM-2203 に DATA WRITE で設定します．
5. VM／VF タイプはジャンパスイッチによりポートアドレスを 088・08 A に変更できます．

4-3-2　YM-2203の制御

4-3-2-1　ROM

インターフェース上のROMのアドレスは，C8000～D7FFF間の16Kバイトです(16Kバイト境界に整列)．既定値CC000～CFFFFを使用します．また，本ROM機能を使用不可能とするKILLスイッチを持っています．

4-3-2-2　割り込み

YM-2203出力の割り込み信号を反転し，IR131またはIR121に接続されます．どちらの信号を利用するかはジャンパスイッチにより設定し，ソフトウェアはその設定をYM-2230のI/OポートAの最上位ビット(IRST0)により判断します．

〝0″の場合がIR131(INT6)，〝1″の場合がIR121(INT5)を使用することを表しています．通常は，IR131(INT6)を使用します．

なお，他機種サウンドボードとのソフトウェア上の互換性を保つため，ソフトウェアは以下のように判断します．

IRSTO	IRST2	割り込み名	信号名
0	I	INT6	IR131
I	I	INT5	IR121
I	0	INT4	IR101
0	0	INT0	IR31

表4-3-2　割り込みポート

注意：

1. リセット時，割り込み信号は〝L″レベルにあるが，ソフトウェアでサウンド用割り込みを使用しない場合は，他の割り込みとの競合を避けるため，割り込み信号を〝H″状態に保つようにしてください(フラグをリセットしないようにする)．
2. 将来の拡張のため，本ボードの割り込み機能をジョイスティックのために使用しないでください．

4-3-2-3　サウンド出力

サウンドボードの外部出力端子にケーブルを接続すると，本体のスピーカーからは本ボードによって生成されたサウンドは出力しません(ただし，VM／VFタイプのLINEOUT1はケーブルを接続しても本体スピーカーから出力します)．BEEP音は，ケーブルの接続にかかわらず本体のスピーカーから出力されます．

本体スピーカーはボリュームを持ちますが，サウンドボードの外部出力端子の出力レベルは固定です．

4-3-2-4　YM-2203

YM-2203のマスタクロックは3.9936 MHzとします．従って，リセット動作を保証するためにマスタクロックの分周数を6以外に設定してください．

4-3-3　サウンドBIOS

PC 9801 U 2／VF／VMによるサウンドボードの利用を容易にするためにBIOSが用意されています．それらの機能は，以下の6つです．

・初期化
・演奏
・演奏の終了
・OPNパラメータの設定／読み出し
・割り込みプロセスの設定
・その他の演奏補助機能

音楽の演奏はハードウェアの割り込みによって実行されますので，他のプログラムと並行して実行できます．サウンドBIOSは，MUSIC BIOSと排他的に存在します．

4-3-3-1　利用者とのインターフェース

サウンドBIOSと上位プログラムとのインターフェースは，各種レジスタ，共通制御情報通知域，パラーメータリストによって行われます．

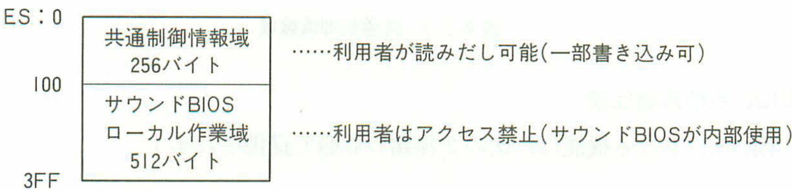


図4-3-2　インターフェース

この他に，次の作業域を必要とします．

・システム共通域(0000：05 E 0 H)の4バイト
・制御情報通知域に続く512バイト
・スタック領域の96バイト

下の表は，共通制御情報域の詳細です．

相対アドレス	フィールド名	サイズ	説　　明
00H 02H 04H 06H 08H	BUF_SEG_n BUF_OFS_n BUF_LNG_n BUF_PTR_n BUF_VDL_n	2 2 2 2 2	PLAYバッファのセグメントベース（Initialize時に設定要） PLAYバッファのオフセットアドレス（　〃　） PLAYバッファサイズ（　〃　） PLAYバッファ内ポインタ PLAYバッファ有効バイト数(バッファ残量)
0AH	BUF_INTC_n	2	PLAYバッファエンプティ割り込み条件 bit15＝{0：未設定 / 1：割り込みEnable}　bit14～0：割り込み発生有効バイト数 (bit15＝0のときは無意味)
0CH	BUF_INT_OFS_n	2	PLAYバッファエンプティ割り込みプロセス オフセット
0EH	BUF_INT_SEG_n	2	PLAYバッファエンプティ割り込みプロセス セグメントベース
10H	KY_n	1	カレントKey-No. 現在発音中のKey-No.　0～60H：発音中 それ以外：発音していない
11H	LN_n	1	デフォルト音長 SET LENGTHにより設定された音長
12H	TCH_n	1	カレントタッチ SET TOUCHにより設定されたG／S値
13H	PLY_n	1	演奏中フラグ 概当チャンネルで演奏中である事を示す 0：非演奏中 FF：演奏中
14H～1FH	SB_WK_1_n	12	サウンドワークエリア1　サウンドBIOSが使用(ユーザー利用不可)
20H～BFH	n＝2～6	160	相対アドレス0～1FまでをCH2～6に関して確保
C0H	TP	1	テンポ数 SET TEMPO により設定したテンポ数
C1H	SAV_KYS	1	Keyステータスセーブエリア ALL STOPによりセーブされたKeyステータス
C2H ～ FFH ～ 2FFH	SB_WK_2 SB_WK_3	48 512	サウンドワークエリア2，3 サウンドBIOSが使用(ユーザー利用不可)

表4-3-3　共通制御情報域

4-3-3-2　BIOS各種共通仕様

サウンドBIOSにおける機能は，次の2種類の形態で提供されます．

リアルタイム機能	実行要求後，直ちに実行される形態
ディレイド機能	PLAYコマンドのデータとして与えられ演奏につれて実行される形態

表4-3-4　BIOSの機能

ディレイド機能は，リアルタイム機能の一部を含んでいますので，実行形態の違いや情報の受け渡し方法を除けば同一の機能を持っています．

また，次のような条件では，サウンドBIOSの動作が保証されません．

・本項で説明する以外の機能コードやパラメータを与えた場合
・サウンドBIOSを介さず直接ハードウェアを使用した場合
・サウンドBIOSの動作を制御するOPNレジスタを不正な値に書き換えた場合

4-3-3-3　各機能の詳細な説明

INITIALIZE

機　能

サウンドボードとサウンドBIOSの初期設定，共通作業域の設定

入　力

AH(機能コード)：00 H

ES（サウンドBIOS作業ベース)：0000 H～FFFFH

注　意

・サウンドBIOS利用時には，このコマンドを必ず最初に実行しなければなりません.

・サウンドBIOSが使用する共通制御情報域のサイズは256バイトで，その後にサウンドBIOSのローカル作業域として512バイトを必要とします.

・利用者は，このコマンドを呼び出す以前にPLAYバッファを確保しておき，そのロケーション・長さをBUF_OFFn, BUF_SEGn, BUF_LNGnフィールドに格納しておかなければなりません(PLAYバッファ長は2の倍数にしてください).

・利用者は，サウンド機能終了まで，共通制御情報域とサウンドBIOSローカル作業域の内容を保証しなくてはなりません.

・INITIALIZEコマンド実行時の各値の初期状態を次に示します.

音長：48 step
テンポ：120(48 step／min)
Gdte time：8
CH 1～CH 3パラメータ：不定
CH 4～CH 6パラメータ：不定

各音源のパラメータは不定のため，INITIALIZE後に設定を行う必要があります.

PLAY

機　能

与えられたディレイド機能，データ列の実行

入　力

AH(機能コード)：10 H

ES／BX(パラメータリストロケーション)……ES：セグメントベースアドレス
BX：オフセットアドレス

注　意

・このコマンドは利用者より与えられたパラメータリストに従い，各チャンネルの演奏を開始します．パラメータリストは，各チャンネルのデータブロックのロケーション，長さを情報として持つ28バイトのデータです．

・このコマンドは各チャンネルのデータブロックよりPLAYバッファに転送された時点で終了し，以降の演奏はインターバルタイマからの割り込みによる演奏ルーチンが行います．

ES：BX+0	データブロックセグメントベース	(未使用)
+4	データブロック1　オフセット	データブロック1　データ長
+8	データブロック2　オフセット	データブロック2　データ長
+C	データブロック3　オフセット	データブロック3　データ長
+10	データブロック4　オフセット	データブロック4　データ長
+14	データブロック5　オフセット	データブロック5　データ長
+18	データブロック6　オフセット	データブロック6　データ長
+1C		

図4-3-3　パラメータリスト(PLAY)

・各データブロックは，ディレイド機能に基づいたメモリブロックです．データブロックの中にディレイド機能以外のデータがあった場合の動作は保証されません．

・データブロックは，INITIALIZE時に設定したバッファの空きエリアよりも短くなければなりません．バッファの空きエリアよりも大きなデータを与えた場合の動作は保証されません．

・利用者が，各チャンネルの演奏の終了を判定するには，バッファエンプティ割り込みを利用するか共通制御情報域を参照して有効バイト数をみる必要があります．

CLEAR

機　能

現在の演奏を中止し，PLAYバッファをクリア

入　力

AH(機能コード)：02 H

AL(処理指定)……00：演奏の中止，バッファクリア
01：演奏の中止，バッファクリアと共通制御情報域の初期化

注 意

・サウンドBIOSの利用を終了する場合には，必ずこのコマンドを呼び出さなくてはなりません．
・このコマンドを実行後，共通制御情報域およびサウンドBIOSローカル作業域を保証する必要はありませんが，INITIALIZEコマンドの実行なしにサウンドBIOSを使用する場合には，保証しなくてはなりません．

READ REG

機 能

指定されたOPNレジスタの内容を読み出す

入 力

AH(機能コード)：10 H

AL (レジスタ番号)：00～FFH

出 力

BH：00 H

BL(レジスタ内容)：00～FFH

注 意

・OPNレジスタのうち，特にFM音源関係のレジスタは読み出し不可能なため，このコマンドとWRITE REGコマンドを使用しなければなりません．
・このコマンドで保証するのはWRITE REGコマンドにより書き込んだ値についてのみであり，ハードウェアに対し直接書き込んだ場合には保証しません．

WRITE REG

機 能

指定されたOPNレジスタに値を書き込む

入 力

AH(機能コード)：11 H

AL (レジスタ番号)：00～FFH

BL (設定値)：00～FFH

＜ディレイド機能フォーマット＞

第1バイト：81 H

第2バイト(レジスタ番号)：00～FFH

第3バイト(設定値)：00～FFH

注 意

・サウンドBIOSを利用する場合は，必ずこのコマンドによってレジスタの内容を変更するようにしてください．READ REGコマンドは，このコマンドとの併用において保証されます．

・このコマンドは,OPNの全レジスタについて書き込みを許可していますが，インターバルタイマ等のサウンドBIOSの基本動作を規定するレジスタを変更する場合，以降の動作については保証されません．

・このコマンドは，なるべく各音源のパラメータの一時的変更の目的に使用するようにしてください．

SET TOUCH

機 能

GATE TIME／STEP TIME値の設定

入 力

AH(機能コード)：12 H

AL(指定チャンネル)……00～05：CH 1～6に対応

AL(G／S値)……00～07 H：1/8～8/8に対応

＜ディレイド機能フォーマット＞

第1バイト：82 H

第2バイト(G／S設定値)：00～07 H

注 意

・NOTEコマンド実行時のGATE TIME／STEP TIME(G／S値)を設定します．以後のNOTEコマンド実行時は，このコマンドの設定値でON／OFFされます．

・このコマンドは，音長の時間に対し，実際に音を出している時間(GATE TIME)を規定します．しかし,STEP TIMEが12以下の音長に対してはBIOS内部で適当に分割するため，必ずしもG／S値で指定した比率とはなりません(近似値をとります)．また，エンベロープ形状の設定状態によっては，小さいG／S値に設定した場合に発音しても聞き取れない場合があります．

※ STEP TIME について

サウンド BIOS では，音長を STEP TIME で指定します．テンポ設定は，この STEPTIME 48 に対する 1 分間の演奏回数で設定されます．通常の演奏は，1 小節あたり 96 STEP TIME で十分ですが，余裕をみて 1 小節 192 STEP TIME とすればほとんどの演奏に不自由を感じることはないでしょう．

1 小節を 192 STEP TIME とすれば，48 STEP TIME は 4 分音符に相当します．テンポ設定コマンド(SET TEMPO)は，楽譜表記の 4 分音符記号＝n における n の値を設定するものと考えることもできます．以下に 1 小節あたりの STEP TIME と各音長の STEP TIME を示します．

音長 I　　II	全音符＝192			全音符＝96		
	通常	付点	3連	通常	付点	3連
2分音符	96	144	64	48	72	32
4分 〃	48	72	32	24	36	16
8分 〃	24	36	16	12	18	8
16分 〃	12	19	8	6	9	4
32分 〃	6	9	4	3	*	2
64分 〃	3	*	2	*	*	1

*整数値では設定不可

表 4-3-5 Step Time

NOTE

機　能

音程・音長の設定

入　力

AH(機能コード)：13 H

AL(指定チャンネル)……00～05 H：CH 1～6 に対応

BH(KEY NO.)……00～60 H：KEY No.. 0～96 に対応

80 H：休符

BL(音長)……00 H：省略

01～FFH：音長(STEP TIME)

<ディレイド機能フォーマット>

第 1 バイト……00～60 H：KEY No.

80 H：休符

第 2 バイト……00 H：省略

00～FFH：音長(STEP TIME)

注 意

・音程・音長の指定に従って発音します．G／S値は，SET TOUCHで設定された値をとります．

・NOTEコマンドで使用されるKEY NO.(0~96)と発音音程の対応は次表のとおりです．

・リアルタイム機能においては，指定チャンネル発音中に実行した場合，前の音が切れて新しい音が発音されます．

・NOTEコマンドの特別な用法以外，ディレイド機能として使用するようにしてください．

	O1	O2	O3	O4	O5	O6	O7	O8	O9
C	0	C	18	24	30	3C	48	54	60
C#(D♭)	1	D	19	25	31	3D	49	55	
D	2	E	1A	26	32	3E	4A	56	
D#(E♭)	3	F	1B	27	33	3F	4B	57	
E	4	10	1C	28	34	40	4C	58	
F	5	11	1D	29	35	41	4D	59	
F#(G♭)	6	12	1E	2A	36	42	4E	5A	
G	7	13	1F	2B	37	43	4F	5B	
G#(A♭)	8	14	20	2C	38	44	50	5C	
A	9	15	21	2D	39	45	51	5D	
A#(B♭)	A	16	22	2E	3A	46	52	5E	
B	B	17	23	2F	3B	47	53	5F	

表4-3-6 KEY NO.と音程の対応表

SET LENGTH

機 能

規定値音長の設定

入 力

AH(機能コード)：14 H

AL(指定チャンネル)……00~05 H：CH 1~6に対応

BL(音長)……01~FFH：STEP TIMEで指定

<ディレイド機能フォーマット>

第1バイト：83 H

第2バイト(音長)……01~FFH：STEP TIMEで指定

注 意

このコマンドは，NOTE，HOLDSTATEにおける規定値音長設定用です．

相対アドレス (Hex) Word	Byte	PADA No. (Dec)	サイズ	フィールド名	説明	FM／SSG
A	5	5	B	OPR_MSK	各オペレータの使用／不使用 MSB LSB × × × × S_4 S_3 S_2 S_1 Sn オペレータ ON／OFF n＝1～4 0：OFF オペレータNo. 1：ON	F
C ～ 12	6 ～ 9	6 ～ 9	B	DC_R_1 ～ 4	各オペレータのディケイ・レート 0～1Fh 短 長	F
14	A	10	B	WAV_FORM_LFO	LFO変調波形 0：ノコギリ波 1：矩形波(デューティ50%) 2：三角波 3：SOH(ランダム)	F／S
16 ～ 1C	B ～ E	11 ～ 14	B	SS_R_1 ～ 4	各オペレータのサスティンレート 0～1Fh 短 長	F
1E	F	15	B	SYNC_DLY_LFO	LFO SYNCディレイタイム 0：非同期 1～0FFh：16ms単位で遅延同期	F／S
20 ～ 26	10 ～ 13	16 ～ 19	B	RL_R_1 ～ 4	各オペレータのリリース・レート 0～0Fh 小 大	F
28	14	20	W	SPEED_LFO	LFO効果の速度 0～3FFFh $\text{LFO周波数(Hz)} = \frac{\text{SPEED_LFO}}{16383} * 250$	F／S
2A ～ 30	16 ～ 19	21 ～ 24	B	SS_L_1 ～ 4	各オペレータのサスティン・レベル 0～0Fh 小 大	F
32	1A	25	B	P_MOD_LFO	LFO効果ピッチ変調の深さ 80～0～7Fh －128～0～127 (逆相)大～小～(正相)大	F／S
34 ～ 3A	1B ～ 1E	26 ～ 29	B	OP_L_1	各オペレータの出力レベル 0～7Fh 小 大	F
3C	1F	30	B	A_MOD_LFO	LFO効果振幅変調の深さ 80～0～7Fh －128～0～127 (逆相)大～小～(正相)大	F
3E ～ 44	20 ～ 23	31 ～ 34	B	KEYSCL_1 ～ 4	KEYスケーリング深さ 0～3 浅 深	F
46	24	35	B	P_MOS_LFO	LFO効果ピッチ変調の深さ(粗調整) 0～0Fh 小 大	F／S

表 4-3-7 SET LENGTH

SET TEMPO

機　能

テンポ(音数)の設定

入　力

AH(機能コード)：15 H

BL(テンポ数)……01~FFH：48 STEP／MIN 単位で指定

<ディレイド機能フォーマット>

第1バイト：84 H

第2バイト(テンポ数)……01~FFH：48 STEP／MIN 単位で指定

注　意

全チャンネル共通にテンポを設定します．テンポ数の単位は1分間に48 STEP TIME の NOTE コマンドで実行できる回数です．あまり速いテンポ数を設定すると，演奏が乱れることがあるので注意してください．

SET PARABLOCK

機　能

各チャンネルのパラメータをまとめて設定する

入　力

AH(機能コード)：16 H

AL(指定チャンネル)……00~05 H：CH 1~6 に対応

ES／BX(設定パラメータロケーション)

DL(パラメータブロックの形式)……00：WORD

01：BYTE

<ディレイド機能フォーマット>

第1バイト：85 H

第2バイト(パラメータブロック形式)……00：WORD

01：BYTE

第3，4バイト(設定パラメータブロック開始オフセット)

第5，6バイト(設定パラメータブロック開始セグメント)

注 意

このコマンドはFM音源チャンネルに対し音色パラメータをブロック指定するものです．サウンドBIOSは，このコマンドを与えられると指定チャンネルの発音を停止し，上位によって示されるアドレスより存在する100／51バイトのパラメータブロックを指定チャンネルに対して設定します．音色パラメータ設定直後は，LFO効果はONになっています．指定チャンネルがFM音源でなかった場合は無視します．

相対アドレス(16進) Word	Byte	パラメータNo. (10進)	サイズ	フィールド名	説明	FM／SSG
0	0	0	B	FB_ALG	オペレータ1の変調度／アルゴリズム MSB LSB × × F_2 F_1 F_0 A_2 A_1 A_0 A_2〜A_0：アルゴリズム0〜7 F_2〜F_0：変調度0〜7	F
2〜8	1〜4	1〜4	B	AT_R_1〜4	各オペレータのアタックレート 0〜1FH 短 長	F
A	5	5	B	OPR_MSK	各オペレータの使用／不使用 MSB LSB × × × × S_4 S_3 S_2 S_1 Sn n=1〜4 オペレータNo. オペレータ ON／OFF 0：OFF 1：ON	F
C〜12	6〜9	6〜9	B	DC_R_1〜4	各オペレータのディケイレート 0〜1FH 短 長	F
14	A	10	B	WAV_FORM_LFO	LFO変調波形 0：ノコギリ波 1：矩形波(デューティ50%) 2：三角波 3：S／H(ランダム)	F／S
16〜1C	B〜E	11〜14	B	SS_R_1〜4	各オペレータのサスティンレート 0〜1FH 短 長	F
1E	F	15	B	SYNC_DLY_LFO	LFO SYNCディレイタイム 0：非同期 1〜0FFH：16ms単位で遅延同期	F／S
20〜26	10〜13	16〜19	B	RL_R_1〜4	各オペレータのリリースレート 0〜0FH 小 大	F
28	14	20	W	SPEED_LFO	LFO効果の速度 0〜3FFFH LFO周波数(Hz)$=\frac{SPEED_LFO}{16383}*250$	F／S

2A〜30	16〜19	21〜24	B	SS_L_1〜4	各オペレータのサスティンレベル 0〜0FH 大 小	F
32	1A	25	B	P_MOD_LFO	LFO効果のピッチ変調深さ 80H〜0〜7FH −128〜0〜127 (逆相)大〜小〜(正相)大	F/S
34〜3A	1B〜1E	26〜29	B	OP_L_1	各オペレータの出力レベル 0〜7FH 小 大	F
3C	1F	30	B	A_MOD_LFO	LFO効果振幅変調深さ 80H〜0〜7FH −128〜0〜127 (逆相)大〜小〜(正相)大	F
3E〜44	20〜23	31〜34	B	KEYSCL_1〜4	KEYスケーリング深さ 0〜3 浅 深	F
46	24	35	B	P_MOS_LFO	LFO効果ピッチ変調深さ(粗調整) 0〜0FH 小 大	F/S
48〜4E	25〜28	36〜39	B	MULT_1〜4	各オペレータのマルチプル 0〜0FH ½〜15倍	F
50	29	40	B	(Rfu)	予約	
52〜58	2A〜2D	41〜44	B	DETUN_1〜4	各オペレータのデチューンレート 0FCH〜0〜03H −4〜0〜3	F
5A	2E	45	B	(Rfu)	予約	
5C〜62	2F〜32	46〜49	B	A_MOS_LFO_1〜4	LFO効果振幅変調深さ(粗調整) 0〜0FH 小 大	F
64	33	50	B	INT_KY_SAV	内部作業用KeyステータスセーブエリアOPNレジスタ♯28に送られたもののコピーが各チャンネル毎にSAVEされている.(内部ワークに保持)	

表4-3-8 SET PARABLOCK

READ PARA

機 能

各チャンネルの各パラメータの値を読む

入 力

AH(機能コード):17H

AL(指定チャンネル)……00〜05:CH1〜6に対応

BL(パラメータ番号)……00〜31H:パラメータNo.0〜49に対応

出　力

BX(設定されている値)：0000～FFFFH

注　意

指定チャンネルの音色パラメータの内容を読みます．パラメータがBYTEの場合，BXの上位バイトはクリアされます．SSG音源について，FM音源のパラメータを指定した場合に返される値は不定です．

WRITE DATA

機　能

各チャンネルのパラメータの設定

入　力

AH(機能コード)：18 H

AL (指定チャンネル)……00～05 H：CH 1～6に対応

BL (パラメータ番号)……00～31 H：パラメータNo.0～49に対応

DX(設定値)：0000～FFFFH

<ディレイド機能フォーマット>

第1バイト：86 H

第2バイト(パラメータ番号)……00～31 H：パラメータNo.0～49に対応

第3，4バイト(設定値)……0000～FFFFH：LOW-HIGHの順

注　意

パラメータがバイトの場合，DLまたは第3バイトの値がセットされます．SSG音源に対して，FM音源のパラメータを設定した場合の動作は保証されません．

ALL STOP

機　能

演奏の一時中断

入　力

AH(機能コード)：19 H

注　意

・全てのチャンネルのON／OFF状態はセーブされ，CONT PLAYコマンド実行に備えます．このコマンドは，インターバルタイマよりの割り込みを禁止するもので，再び割り込みをスタートさせるには，PLAYコマンドかCONT PLAYコマンドを実行してください．
・このコマンドによる演奏停止は，ハードウェア制御割り込み単位で行われます．
・CONT PLAYで再開する場合，共通制御情報域およびサウンドBIOSローカル作業域は保証されなければなりません．

CONT PLAY

機　能

ALL STOPにより中断した演奏の再開

入　力

AH(機能コード)：1AH

注　意

ALL STOPによりセーブされた各チャンネルのON／OFF状態は復旧され割り込みが再開されます．ALL STOPが行われていないのにCONT PLAYを行った場合には，BIOSの動作には何も影響を与えません．

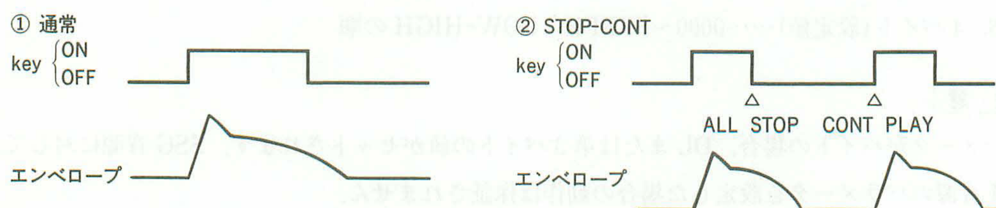


図4-3-4　一時停止と再開のタイミング

エンベロープはサウンドBIOSで直接制御できないため，KEY-ONのタイミングでスタートします．ALL-STOPの後，再開する必要がなければ，いかなるマクロを実行しても構いません．

HOLD STATE

機　能

KEY ON／KEY OFF の状態を維持する

入　力

AH(機能コード)：1EH

AL (指定チャンネル)……00～05 H：CH 1～6 に対応

BL (維持する長さ)……00：省略(規定値)

　　　　　　　　　　01～FFH：STEP TIME

<ディレイド機能フォーマット>

第 1 バイト：89 H

第 2 バイト(維持する長さ)……00：省略(規定値)

　　　　　　　　　　　　　01～FFH：STEP TIME

注　意

- このコマンドは，KEY ON／KEY OFF の状態を変更した後に，一定時間その状態を保つ場合に使用します．
- NOTE コマンドにより休符を指定した場合は，最初に KEY OFF を行うためこのコマンドのようには使えません．NOTE コマンドの後にこのコマンドを使用すれば，休符と同じ効果になります．

MODU ON

機　能

LFO 効果の ON

入　力

AH(機能コード)：1 BH

AL (指定チャンネル)……00～05 H：CH 1～6 に対応

<ディレイド機能フォーマット>

第 1 バイト：87 H

注　意

このコマンドは，LFO パラメータの規定値が何であっても影響なく実行されます．つまり，本来 LFO 効果のない音色であっても LFO 効果を付けようとします．しかし，実際に設定されている LFO パラメータが全て 0 ならば効果はあらわれません．

MODU OFF

機 能

LFO効果のOFF

入 力

AH(機能コード):1CH

AL(指定チャンネル)……00~05H:CH1~6に対応

<ディレイド機能フォーマット>

第1バイト:88H

SET INTC OND

機 能

PLAYバッファエンプティ割り込みの条件を設定

入 力

AH(機能コード):1DH

AL(指定チャンネル)……00~05H:CH1~6に対応

ES/BX(割り込みプロセスエントリ)

CX(有効バッファ長)……BIT15=1:割り込みDISABLE

BIT15=0:割り込みENABLE

BIT14~0:有効バッファ長

注 意

・指定チャンネルの有効バイト数が指定値以下になった時,割り込みプロセスをFAR CALLします.

・有効バイト数指定値のBIT15が0の時は,BIT14~0の値は意味をもちません.

・割り込みプロセスは,INTERRUPT FLAGがOFFの状態でCALLされます.

・割り込みプロセス中では,外部割り込みはできないようにしておいてください.割り込みプロセスの実行時間はできるだけ短くしてください.あまり長いと演奏のテンポが狂うなどの障害が起こる場合もあります.

・割り込みプロセスエントリ時のレジスタの内容は以下のとおりです.

AX:割り込み発生チャンネルNo.

BX:有効バイト数

それ以外のレジスタについては不定ですが，利用者が割り込みプロセス内で保証する必要はありません．



図 4-3-5　割り込みのタイミング

SET VOLUME

機　能

FM 音源の VOLUME の設定

入　力

AH(機能コード)：1 FH

AL (指定チャンネル)……00～02 H：CH 1～3 に対応

BL (設定値)：00～7 FH

＜ディレイド機能フォーマット＞

第 1 バイト：8 AH

第 2 バイト：00～7 FH

注　意

・設定値は増加値表現で 00 H が最小，7 FH が最大です．

・このコマンドはコマンド実行時に対応するチャンネルのパラメータフィールド(FB_ALG)を参照し，キャリアオペレータ(最も出力に近いオペレータ)の音量を操作します．

4-3-4　サウンド BIOS 使用方法

4-3-4-1　INT ベクタの設定

サウンド BIOS 各機能の呼び出しは，内部 INT コールによります．利用者は前もって割り込みベクタを設定する必要があります．

そのための情報は，実メモリアドレスの CEE 00 H より以下の形式で格納されています．

(未定) H+0

0 I H			
D 2 H	0 0 H	サウンドBIOSエントリ	→CEEO：0のオフセット

図4-3-6　INTベクタの設定

利用者は，この情報をもとに適当なINT番号に割り当ててください.

サウンド演奏のために使用される割り込みベクタは(INT 14 HまたはINT 15 H)INITIALIZE時にBIOSが呼び出して決定します.

4-3-4-2　呼び出し方法

サウンドBIOSの呼び出しは，前述したINT番号をもとに内部INTで行います．この時のSS，SPはサウンドBIOSで使用可能なスタックアドレスをセットしておいてください.

4-3-4-3　リターン条件

特に規定のないものについては全て保証します.

4-3-4-4　注意

サウンドBIOS処理中においては，外部割り込みが可能となっています.

4-3-5　サウンド拡張BASIC

アセンブラ言語等を使ってサウンドBIOSに割り込みをかけなくても，N 88-DISKBASIC version 3.0，N 88-日本語BASICインタプリタ／コンパイラ(MS-DOS版)を使用することによって，簡単なプログラムであれば音楽演奏のために拡張されたBASICによって記述することができます.

コマンド名	機　　　能
PLAY ALLOC	サウンドバッファの確保および初期化
PLAY	音楽演奏
VOICE	音色バンクの利用者定義
VOICE COPY	音色バンクの音色パラメータをコピー
VOICE LFO	各チャンネルの出力にLFO効果付加
VOICE REG	シンセサイザLSIのレジスタに値を設定
VOICE INIT	音色バンクの初期化
ON PLAY(c, n)GOSUB	PLAY割り込み処理ルーチンの開始行の定義
PLAY ON／OFF／STOP	PLAY割り込みの許可／禁止／停止を設定
STATUS PLAY(C)	サウンドバッファの未演奏データのバイト数の調査

表4-3-9　拡張サウンドBASICコマンド一覧

これら，BASICの拡張命令を使って簡単なプログラムを作成してみました.

プログラムは，ある書式に従って作成した演奏データファイルを読み込み，各処理ルーチンを呼び出すものです．以下に，使用法と演奏データファイルの書式について説明します．

4-3-5-1 使用法

プログラムは，N 88-DISKBASIC(86) version 3,0，もしくはN 88-日本語 BASIC(86) version 3.0 MS-DOS版のどちらでも動作しますが，メモリスイッチなどの設定をPC 9801-26(ミュージックボード)のマニュアルを見て変更しておいてください.

なお，MS-DOS版のN 88-日本語 BASIC(86) version 3.0を使用する場合には，起動の際に，

```
A>N 88 BASIC ／E：SOUND
```

というように，拡張命令を追加する引数をとる必要があります.

BASICが起動したら，プログラム 4-3-1を実行してください．演奏データファイルのファイル名を入力するように促されますので，プログラム 4-3-2のファイル名を入力してください(ただし，プログラム 4-3-2はディスクにアスキーセーブされている必要があります)．プログラム 4-3-2の読み込みが終ると，自動的に演奏が開始されます.

プログラムを停止させる場合は，STOPキーを押してください．再度，演奏プログラムを実行したい場合は，もう一度ディスクからプログラム 4-3-1を読み込んで実行して下さい.

4-3-5-2 演奏データファイルの書式

演奏データファイルは，3つのサブルーチンによって構成されています.

● INIT……FM音源が使用するデータエリア・変数を初期設定

CLEAR文によって，FM音源のデータエリアを確保しますが，DISK-BASICを使用する場合とMS-DOS版のBASICを使用する場合とでは，引数の取り方に違いがあります．また，このプログラムで確保しているデータエリアの大きさは，3音源を使用する場合のものです．PLAY ALLOC文，DIM文の引数の値も，何の音源を使用するかによって変えなければなりません．サブルーチンでCLEAR文を実行しているので，復帰するためにGOTO文を使用しています.

● GRAPH……バックグラウンド演奏中に表示するグラフィックサブルーチン

プログラムは簡単なもので構いませんが，必ず無限ループにしてください.

● PLAY.MUSIC……実際に演奏を行うサブルーチン

演奏データは，BASICのDATA文によって与えられ，バックグラウンド演奏されます．演奏データは，MML(ミュージック・マクロ・ランゲージ)の書式に基づいて書かれます．このプログラムは，バックグラウンド演奏を目的としていますので，必ずデータの先頭を〝MB〟としてください.

コマンド名	書式	機能	備考
Ax~Gx	A~G〔〈半音〉〕〔〈音長〉〕	音程，音長を指定する	
Mx	M〈周期〉	SSGエンベロープ周期を設定する	SSG
Sx	S〈形状〉	SSGエンベロープ形状を設定する	SSG
Vx	V〈音量〉	音量を設定する	
Lx	L〈音長〉	デフォルト音長を認定する	
Qx	Q〈割合〉	ゲートタイム／ステップタイム比を設定する	
Ox	O(オクターブ値〉	デフォルトのオクターブ値を設定する	
>	>	デフォルトのオクターブ値を1つ上げる	
<	<	デフォルトのオクターブ値を1つ下げる	
Kx	K〈キー番号〉	キー番号に対する音階指定	
Tx	T〈テンポ値〉	テンポを設定する	
Rx, Px	R(P)〔〈音長〉〕	休符を指定する	
+(#), −	(音程){+ # −}	半音上下の記号	
.(ピリオド)	(音長).	符点音符の記号	
{……} x	{(MML)}〈音長〉	連符	
@x	@〈音色番号〉	音色番号の指定	FM
Yr, d	Y〈レジスタ番号〉,〈数値〉	シンセサイザLSIレジスタへの書込	
@Vx	@V〈音長〉	FM音源の音量を細かく設定する	FM
@Wx	@W〈音長〉	一定時間なにもせずに待つ	
_(アンダーバー)	_〈音程〉	移調を指示する	
Zp, v	Z〈パラメータ番号〉,〈数値〉	バックラウンド指定	
!	!	LFO効果の一時的ON	
*	*	LFO効果の一時的OFF	
MB	MB	バックラウンド指定	
MF	MF	フォアグラウンド指定	
=x;	=〈変数〉;	数値の間接指定	
X=x;	X=〈文字変数〉;	MMLの文字列による置換	

表 4-3-10 MML の一覧表

音色番号	音色名	音色番号	音色名	音色番号	音色名
0	DEFAULT	28	EPIANO3	56	RAIN DROP
1	BRASS2	29	GUITAR	57	HORN
2	STRING2	30	EBASS1	58	SNARE DRUM
3	EPIANO3	31	EBASS2	59	COW BELL
4	EBASS1	32	EORGAN1	60	PERC1
5	EORGAN1	33	EORGAN2	61	PERC2
6	PORGAN1	34	PORGAN1	62	PERC3
7	FLUTE	35	PORGAN2	63	MUSIC BOX
8	OBOE	36	FLUTE	64	CELLO
9	CLARINET	37	PICCOLO	65	LOW BRASS
10	VIBRAPHN	38	OBOE	66	WW ENSNBL
11	HARPSIC	39	CLARINET	67	AC GUITAR
12	BELL	40	GROCKEN	68	FLUTE／HARP
13	PIANO	41	VIBRPHN	69	FUNK PLUC
14	MUSHI	42	XYLOPHN	70	FUNK BASS
15	DESCENT	43	KOTO	71	SYN LEAD
16	UFO	44	ZITAR	72	METAL CLES
17	GRANPRI	45	CLAVINET	73	STAIN
18	LASER1	46	HARPSIC	74	CUBIN GRASS
19	LASER2	47	BELL	75	HUMAN
20	SIN WAVE	48	HARP	76	WOOD BASS
21	BRASS1	49	BELL／BRASS	77	CHIMES
22	BRASS2	50	HARMONICA	78	SPACY
23	TRUMPET	51	STEELDRUM	79	OBOE／B.CLAR
24	STRING1	52	TIMPANI	80	OLD STRING
25	STRING2	53	TRAIN	81	STEEL'S CRY
26	EPIANO1	54	AMBULANCE		
27	EPIANO2	55	TWEET		

表 4-3-11 プリセット音源の一覧表

プログラム 4-3-1 TK-FM.BAS

```
1000 ' TK-FM.BAS ***************************************************
1010 '            PC-Techknow98V FM MUSIC PLAYER version 1.0
1020 '        1986.09.27 Programed by Benny copyright (C) DMSC
1030 '*************************************************************
1040 '
1050 CONSOLE 0,25,0,1 : SCREEN 3,0,0,1 : WIDTH 80,25 : CLS 3
1060 ON STOP GOSUB *TOMARE : STOP ON
1070 LOCATE 0,10
1080 PRINT "                    PC-Techknow98V FM MUSIC PLAYER version 1.0"
1090 PRINT "                      Programed by Benny copyright (C) DMSC"
1100 PRINT "                            use PC-9801-26 SOUND BOARD"
1110 PRINT ""
1120 '
1130 '                                演奏・グラフィック処理のプログラム読み込み
1140 LOCATE 20,14 : COLOR 6 : PRINT "MUSIC DATA FILE NAME :";
1150              COLOR 7 : INPUT FL$
1160 CHAIN MERGE FL$,1170,ALL
1170 '                                各処理の呼び出し
1180 GOTO *INIT
1190 *MODORI
1200 ON PLAY (3,0) GOSUB *PLAY.MUSIC : PLAY ON
1210 GOSUB *PLAY.MUSIC
1220 GOTO *GRAPH
1230 END
1240 '
1250 *TOMARE
1260 '
1270 LOAD "TK-FM.BAS" : END
```

プログラム 4-3-2 ELISE.TFD

```
10000 *INIT
10010  CLEAR ,&H1FDF-(16*3) : ' for DISK-BASIC
10020 'CLEAR 16*3+32        : ' for MS-DOS
10030  PLAY ALLOC 255,255,255
10040  DIM OTO$(3)
10050  GOTO *MODORI
10060 '
20000 *GRAPH
20010 '
20020  SCREEN 3,0,0,1 : CLS 3
20030  X=100 : Y=100 : XX=540 : YY=300 : BX=5 : BY=5 : AX=-15 : AY=-2
20040  CIRCLE(X,Y),5,7 :'CIRCLE(X,Y),5,0
20045  CIRCLE(XX,YY),5,2 :'CIRCLE(XX,YY),5,0
20050  X=X+BX : Y=Y+BY
20055  XX=XX+AX : YY=YY+AY
20060  IF X<0  OR X>639  THEN BX=BX*-1
20065  IF XX<0 OR XX>639 THEN AX=AX*-1
20070  IF Y<0  OR Y>399  THEN BY=BY*-1
20075  IF YY<0 OR YY>399 THEN AY=AY*-1
20080  GOTO 20040
20090 '
30000 *PLAY.MUSIC
30010  PLAY STOP : CLS 2
30020  READ OTO$(1),OTO$(2),OTO$(3)
30030  IF M$(1)="$" THEN RESTORE *MUSIC.DATA
30035  READ OTO$(1),OTO$(2),OTO$(3)
30040  PLAY OTO$(1),OTO$(2),OTO$(3)
```

```
30050 PLAY ON
30060 RETURN
30070 '
30080 *MUSIC.DATA
30090 DATA MBT80@63V1406L16
30100 DATA MBT80@63V1405L16
30110 DATA MBT80@63V1404L16
30120 '
30130 DATA >ED+ ED+E<B>DC <A8R CEA   B8 R EG+B >C8 R<E>ED+ ED+E<B>DC
30140 DATA  RR  RR R R RR <A>EARRR <E>EG+RR R <A>EA R RR  RR R R RR
30150 DATA >ED+ ED+E<B>DC <A8R CEA   B8 R EG+B >C8 R<E>ED+ ED+E<B>DC
30160 '
30170 DATA <A8 R CEA B8  R E>C<B  A4   >ED+ ED+E<B>DC <A8 R CEA  B8 R EG+B
30180 DATA <A>EA RRR <E>EG+R R R <A>EA RRR  RR R R RR <A>EA RRR <E>EG+RR R
30190 DATA <A8 R CEA B8  R E>C<B  A4   >ED+ ED+E<B>DC <A8 R CEA  B8 R EG+B
30200 '
30210 DATA  >C8 R<E>ED+ ED+E<B>DC <A8 R CEA B8  R E>C<B  A8 RB>CD E8. <G>FE
30220 DATA  <A>EA R RR  RR R R RR <A>EA RRR <E>EG+R R R <A>EAR RR CG>C R RR
30230 DATA  >C8 R<E>ED+ ED+E<B>DC <A8 R CEA B8  R E>C<B  A8 RB>CD E8. <G>FE
30240 '
30250 DATA  D8.<F> ED C8.<E>DC <B8R E>E R RE>ERR<D+ ERR D+
30260 DATA  <<G>GBRRR <A>EARRR <E>E>ERRE  >ERRD+ER  RD+ER
30270 DATA  D8.<F> ED C8.<E>DC <B8R E>E R RE>ERR<D+ ERR D+
30320 '
40000 DATA $,$,$
```

プログラム4-3-2は，ベートーベン作曲の「エリーゼのために」をアレンジし，ミュージックボードで演奏するMMLデータです.

4-4 RS-232 C を使った簡単なターミナルプログラム

4-4-1 コンピュータ・ネットワーク

電話回線を利用したパーソナルコンピュータによる通信も，パーソナルコンピュータの利用法として定着してきたように思えます．コンピュータネットワークを実現するためには，パーソナルコンピュータの他にモデム，音響カプラ，RS-232 C ケーブル等の周辺機器類と，ターミナルソフトが必要です.

この節では，通信用インターフェースとして最も一般的な RS-232 C インターフェースを利用して，C 言語による簡単なターミナルプログラムを紹介します.

4-4-2 ターミナルソフト

この項では，ターミナルソフト〝TERM〟について説明します.

実際にこのターミナルソフトではどのように通信を行っているのか，ルーチンごとに細かく説明します. PC-9801 特有の機能はまったく使っていません．従って PC-9801 シリーズ以外のどの MS-DOS マシンでも，実行することが可能です.

4-4-2-1 メインルーチン：main()

プログラムは大きく二つの部分に分けられます．

まず1つは，通信を行う前の準備部分です．タイトルの表示，RS-232 Cの初期化，各種フラグの設定を行っています．

もう1つは，ターミナルソフトのメインともいえるRS-232 Cとの基本的なやり取りを行う部分です．RS-232 Cから送られてきたデータの中からLF(0 AH)，CR(0 DH)，TAB(09 H)，BS(08 H)，ESC(1 BH)以外のコントロールコードを無視し，文字の場合は画面に表示します．キーボードから入力があった場合，そのコードをRS-232 Cへ送ります．ただし，ESC(1 BH)が入力された場合にはコマンド処理ルーチンを呼び出します．

4-4-2-2 コマンド処理ルーチン：command( )

メインルーチンでESC(1 BH)が入力されたときに呼び出されるルーチンです．プロンプト(>>)を表示後コマンドの入力待ちになります．入力されたコマンドはworkという文字型の配列にストアされ，work[0]の内容をもとにswitch~case文で各コマンド処理に分けられます．各処理の説明をします．

●ダウンロード：case 'D'

work[1]がnull，つまり入力が一文字だけの場合ダウンロード中のファイルをクローズし，ダウンロードフラグ(downf)をリセットします．すでに存在するファイルであった場合は新しく作るかアペンドするか尋ねます．その後ダウンロードフラグを立ててターミナルモードへ戻ります．

●アップロード：case 'U'

ダウンロードと同様にwork[1]がnull，つまり入力が一文字だけの場合アップロード中のファイルをクローズし，アップロードフラグ(upf)をリセットします．存在しないファイル名が指定された場合はエラーを表示後ターミナルモードへ戻ります．それ以外の場合ではアップロードフラグを立ててターミナルモードへ戻ります．

●ターミナルソフトの終了：case 'Q'

MS-DOSのコマンドレベルへ戻ります．

●チャイルドプロセスの実行：case '!'

COMMAND.COMをチャイルドプロセスとして呼び出します．

●エスケープコードの送信：case 'B'

エスケープコード(1 BH)がRS-232 Cに送信されます．ターミナルモードでキーボードからエスケープコードが入力されることによってこのプログラムではコマンドモードへ移行してしまいます．したがって，このコマンドによりエスケープコード(1 BH)を送信します．

●ローカルエコーの ON／OFF：case 'L'

work[3]，つまり入力されたコマンド文字列の4文字目が〝N〟であった場合ローカルエコーフラグ(localf)をセットし，〝F〟であった場合リセットします．

● CTRL＋C コードの送信：case 'B'

CTRL＋C コード(03 H)が RS-232 C に送信されます．

4-4-2-3　ダウンロード処理ルーチン：download( )

引数として渡された文字をダウンロードフラグが真のとき，ディスクファイルへの書き込みを行います．

4-4-2-4　アップロード処理ルーチン：upload( )

アップロードフラグ(upf)が真のときディスクファイルから一文字取り出し，引数として返します．ディスクファイルが空になった場合，画面にその旨を表示し，ファイルをクローズします．アップロードフラグが偽のときと，ディスクファイルが空になったときは引数に0を返します．

4-4-2-5　ウェイトルーチン：wait( )

タイミングを取るために一定の時間，処理を停止させるためのルーチンです．

4-4-3　プロトコルの設定

コンピュータ・ネットワークにアクセスするためには様々な取り決めがあることは前項で説明しました．通信する規格，プロトコルの設定はターミナルソフトを実行する前にしておく必要があります．

ここでは，プロトコルの設定を SPEED.COM によって行います．例えば 300 bps・ノンパリティ・8ビット長・ストップビット1・X コントロールあり，といった設定の場合には以下のようにキーボードから打ち込みます．それぞれのパラメータの意味は MS-DOS のマニュアルを参照してください．

```
A>SPEED RS232C-0 300 BITS-8 PARITY-NONE STOP-1 XON
```

プロトコルの設定は，それほど変わることがないのでバッチファイルを用意しました．上記のプロトコルの場合は次のバッチファイルを使うと便利です．ファイル名〝T 300.BAT〟でセーブしておけば MS-DOS のコマンドレベルで T 300 と入力することによって自動的にプロトコルを設定し，ターミナルソフトを実行します．

プログラム 4-4-1　バッチファイル(E 300.BAT／T 300.BAT／T 1200.BAT)

```
SPEED RS232C-0 300 BITS-7 PARITY-EVEN STOP-1 XON
TERM
```

```
SPEED RS232C-0 300 BITS-8 PARITY-NONE STOP-1 XON
TERM
```

```
SPEED RS232C-0  1200 BITS-8 PARITY-NONE STOP-1 XON
TERM
```

4-4-4　ターミナルソフトの使用法

ターミナルソフトを起動させるとオープニング画面が表示されターミナルモードに入ります．ここで，キーボードから何か文字を入力すればRS-232 Cへ送られ，RS-232 Cから送られてきた文字は画面上に表示されます．こういった基本的な入出力以外にいくつかの機能が用意されていますので，それらを説明します．

ターミナルモードでESCキー(エスケープ・キー)を押すことによりコマンド・モードに移ります．コマンド・モードで使える主なコマンドを以下に示します．コマンドの実行後，もしくはコマンド実行中にエラーが発生した場合はターミナルモードへ戻ります．

? 　メニューリスト

コマンド・モードで使えるコマンドのリストを画面に表示します．コマンドの使い方などを忘れてしまった場合などに利用できます．次のようなメニューが表示されます．

```
  ===== command =====
d  file-name    down-load            (ダウンロード)
u  file-name    up-load／exc-file    (アップロード)
q               Quiet                (終了)
!  dir          invoke command.com   (子プロセスの実行)
e               send escape          (ESCの送信)
l  on／off      local echo           (ローカルエコー)
h               send ^C char.        (CTRL+Cの送信)
```

d　ダウンロード

ダウンロードとは通信内容をディスクに記録させることをいいます．コンピュータ・ネットワークのやりとりをダウンロードすることにより，重要なデータや記事の読み落しを防いだり，エディタやワープロなどで編集したりすることができます．

u　アップロード

アップロードとはディスクに記録されている文章やデータをコンピュータ・ネットワークへ送り出すことをいいます．ダウンロードとは逆にエディタやワープロなどで編集した文章や，プログラムにより作成された表などを送ることができます．これにより，文章をきれいに編集して送ることができます．

!　MS-DOSコマンドの実行

dir，typeといったMS-DOSの内部コマンドおよび外部コマンドを実行する場合に使用します．詳しくはMS-DOSのマニュアルを参照してください．

l　ローカルエコーのON／OFF

キーボードから打った文字をそのまま画面へ表示することをローカルエコーと呼びますが，それをするかどうか決定するのがこのコマンドです．多くのコンピュータ・ネットワークでは受信した内容を再びターミナル側へ返していますが，中には受け取った内容をそのまま返さずに処理してしまうコンピュータ・ネットワークも存在します．この場合，キーボードから打った文字はまったく画面に表示されないため，正しく文字を打ち込んでいるのか確認ができません．このようなときにこのコマンドでローカルエコーをさせれば，問題はなくなります．

4-4-5　プログラムのコンパイルの方法

TERM.Cは，HI-TECH社製のHI-TECH C version 2.3によってコンパイルし実行することが可能です．DRV.ASは，HI-TECH Cに添付されているアセンブラによって記述されています．コンパイルまでの手順は，以下に示したハードコピーを見て下さい．

```
B>DIR TERM.C

 ドライブ B: のディスクのボリュームラベルは TECHKNOW98V
 ディレクトリは B:¥SEC4

TERM     C        4198  86-09-20    4:14
         1 個のファイルがあります.
   604160 バイトが使用可能です.

B>DIR DRV.AS

 ドライブ B: のディスクのボリュームラベルは TECHKNOW98V
 ディレクトリは B:¥SEC4

DRV      AS       1122  86-09-20    4:16
         1 個のファイルがあります.
   604160 バイトが使用可能です.

B>C TERM.C DRV.AS
HI-TECH C COMPILER (MS-DOS) V2.3
Copyright (C) 1984, 1985 HI-TECH SOFTWARE
TERM.C
DRV.AS
```

```
B>DIR TERM

 ドライブ B: のディスクのボリュームラベルは TECHKNOW98V
 ディレクトリは B:¥SEC4

TERM     C      4198  86-09-20   4:14
TERM     OBJ    5499  86-09-22   3:15
TERM     EXE    6840  86-09-22   3:16
        4 個のファイルがあります.
  584704 バイトが使用可能です.
```

画面 4-4-1 TERM. TXT

プログラム 4-4-2 TERM.C

```
/*
 *    PC9801用 TERMINAL PROGRAM
 *    1986年8月1日  Programmed by BOGY.
 */

#include "stdio.h"
#include "ctype.h"
#include "signal.h"

#define         ERROR   -1
#define TRUE    1
#define         FALSE   0
#define         NULL    0

/*
 *    文字を大文字にして返す MACRO
 */
#define         upper(x)(('a'<=x && x<='z')?x-('a'-'A'):x)

staticFILE      *fpd,*fpu;
staticint       downf=0;/* down-load flag */
staticint       upf=0;          /* up-load flag */
staticint       end_flag=1;
staticint       localf;         /* local echo is off */

main()
{
        int     chr,f;
        void    put_c();

        printf("%c*Terminal Program V3.0 for MS-DOS.¥t¥t(C) DMSC¥n",27);
        printf("¥t¥t¥t¥t¥tPrograned by BOGY !!¥n");

        printf("¥nok!¥n");

        signal(SIGINT,put_c);

        rs_init();
        end_flag=TRUE;
        localf=0;

        while(end_flag) {
                while ( (f=rs_stat()) ) {
                        if (f==0xffff) {
                                putch('e');
                                break;
```

```
                        }
                        chr=rs_get();
                        if (  chr>=' ' || chr==0xa || chr==0xd ||
                                                chr==0x9 || chr==0x8 || chr==0x1b ) {
                                putch(chr);
                                download(chr);
                        }
                }
                chr=0;
                if ( kbhit() || (chr=upload())!=0) {
                        if (chr==0) {
                                chr=getch();
                        }
                        if (chr==0x1b) {
                                command();
                        }
                        else {
                                if (chr != 0xa) {
                                        rs_put(chr);
                                }
                                if (localf==1) {
                                                putch(chr);
                                                if (chr==0xd) {
                                                        putch(0xd);
                                                        putch(0xa);
                                                        wait();
                                                }
                                }
                        }
                }
        }
}

/*
 *      コマンド処理ルーチン
 */

command()
{
        char    work[80],c,*p;
        int     chr;

        printf("\n>>");
        p=work;
        while(1) {
                while(( chr=upload() )==0 && !kbhit()) ;
                if ( chr == 0xd ) {
                        chr=upload();
                        break;
                }
                if ( chr == 0 )   {
                        chr=getch();
                        if (chr==0xd) break;
                }
                if (chr==8) if ( p!=work ) p--; else ;
                else *(p++)=chr;
                putch(chr);
        }
        *p=0;
        printf("\n");

        switch(upper(work[0])) {
                case 'D':       /* ダウンロード処理 */
```

```
		if (downf) {
			fclose(fpd);
			downf=0;
			if (work[1]==0) {
				printf("¥n");
				break;
			}
		}
		if ((fpd=fopen(work+2,"r"))==NULL) {
			if (creat(work+2,0)==ERROR) {
				printf("ERROR¥n");
				break;

			}
		}
		else {
			fclose(fpd);
			printf("File alread exist. N)ew or A)ppend ?");
			c=upper(getch());
			printf("¥n");
			if (c=='N')
				if (creat(work+2,0)==ERROR) {
					printf("ERROR¥n");
					break;
				}
		}
		if ((fpd=fopen(work+2,"a"))==NULL) {
			printf("ERROR¥n");
			break;
		}
		downf=1;
		printf("Down-load.¥n");
		break;

	case 'U':	/* アップロード処理 */
		if (upf) {
			fclose(fpu);
			upf=0;
			if (work[1]==0) {
				printf("¥n");
				break;
			}
		}
		if ((fpu=fopen(work+2,"r"))==0) {
			printf("ERROR¥n");
			break;
		}
		upf=1;
		printf("Up-load.¥n");
		break;

	case '?':	/* ＨＥＬＰ処理 */
		printf("¥n ===== command =====¥n¥n");
		printf("d file-name			down-load¥n");
		printf("u file-name			up-load/exec-file¥n");
		printf("q				Quit¥n");
		printf("! dir				invoke command.com¥n");
		printf("e				send escape¥n");
```

```
                        printf("l on/off         local echo¥n");
                        printf("b                       send ^C char.¥n");
                        break;

                case 'Q':       /* 終了処理 */
                        exit(0);
                        break;

                case '!':       /* ＣＯＭＭＡＮＤ．ＣＯＭを呼び出す。*/
                        if (work[1]==0)
                                spawnl("COMMAND.COM",0);
                        else    system(work+2);
                        printf("¥n");
                        break;
                case 'E':       /* ＥＳＣコードを送る */
                        rs_put(0x1b);
                        printf("¥n");
                        break;

                case 'B':       /* break ｷｬﾗ ｦ ｵｸﾙ */
                        rs_put(0x3);
                        printf("¥n");
                        break;

                case 'L':       /* local echo control */
                        switch(upper(work[3])) {
                                case 'N': localf=1; break;
                                case 'F': localf=0; break;
                                default:
                                        printf("ERROR¥n");
                                        break;
                        }
                        printf("¥n");
                        break;

                default:printf("ERROR¥n");
                        break;
        }
        printf("ok!¥n");
}

/*
 *      ファイルへのダウンロード処理
 */

download(c)
char    c;
{
        if (downf) {
                fputc(c,fpd);
        }
        return;
}

/*
 *      ファイルからのアップロード処理
 */

upload()
{
        int     c;
        if (upf) {
                if (EOF==(c=fgetc(fpu))) {
                        upf=0;
```

```
                        printf("¥nUPload. completed.¥n");
                        fclose(fpu);
                }
                else return((c=='¥n')? '¥r' : c );
        }
        return(0);
}

wait()
{
        int     i;
        for(i=0; i<3000; i++);
        return;
}

put_c()
{
        rs_put(0x3);
        return;
}
```

プログラム 4-4-3 DRV.AS

```
        .title HI-TECH C: DRV.AS
.psecttext
        .globl  _rs_init

; INIT

_rs_init:
.globlsmall_model
        push    si
        push    di
        push    bp
        mov     bp,sp
; AUX: cocked -> law
        mov     ax,#4400h
        mov     bx,#3    ; AUX
        int     #21h
        jnc     1f
2: ; error
        mov     ax,#0ffffh
        br      l1
1:
        and     dx,#00ffh
        or      dx,#20h
        mov     bx,#3    ; AUX
        mov     ax,#4401h
        int     #21h
        jc      2b

        mov     ax,#0
l1:
        mov     sp,bp
        pop     bp
        pop     di
        pop     si
        ret

; RS STAT
```

```
        .globl  _rs_stat
_rs_stat:
        push    si
        push    di
        push    bp

        mov     ax,#4406h
        mov     bx,#3
        mov     cx,#1
        int     #21h
        jnc     2f
; error
        mov     ax,#0ffffh
        br      12
; non-error
2:
        mov     ah,al
        or      ax,ax
        brz     12
; data ready
        mov     ax,#1
12:
        pop     bp
        pop     di
        pop     si
        ret

; rs232c 1 char get routine
        .globl  _rs_get
_rs_get:
        push    si
        push    di
        push    bp

        mov     ax,#0300h
        int     #21h
        and     ax,#00ffh

        pop     bp
        pop     di
        pop     si
        ret

; rs232c 1char put routine
        .globl  _rs_put
_rs_put:
        mov     bx,sp
        push    si
        push    di
        push    bp

        mov     dx,2[bx]
        mov     ax,#0400h
        int     #21h

        pop     bp
        pop     di
        pop     si
        ret
```

Vシリーズによるを考える 第5章

5-1 Vシリーズで新世代を体験する

5-1-1 ワークステーションの概念

ワークステーションとは作業をする場であり，目的の仕事を行うのに最適な環境を提供するものが理想です．プログラム開発をするためのソフト開発ワークステーション，AI 研究用の AI ワークステーション等があります．

PC-9801 も考え方によっては一種のワークステーションとすることができます．市販の高価なワークステーションとは違って機能は制限されますが，個人用としては，それなりの役割を果してくれると思われます．特徴的なのは，

・ビットマップディスプレイ

・マルチ・ウィンドウ

・高度な作業環境

です．いずれも強力なコンピューテーションパワーと大量のメモリ空間を必要とします．

5-1-2 ビットマップ・マルチウィンドウ

ここでは，PC-9800 シリーズ上で C 言語を使用して，簡単なウィンドウ・ライブラリ「Tech-Window-System」を作成してみました．ライブラリは Lattice C ver 3.0，及びアセンブラで記述されていますので，ユーザーアプリケーションから関数として呼び出すことによって，マルチウィンドウシステムが構成されます．このシステムの制作には，ライフボード社製 C-TOOL／98 を使用しました．

5-1-2-1 全体構成

マルチウィンドウシステムの設計で，まず最初に決定しなければならないことは，文字，ビットマップの扱い方です．文字もフォントを使ってビットマップとする方法が一般的ですが，ここでは文字を TEXT-VRAM に，ウィンドウ等のビットマップ部を G-VRAM に書き込む混合型としました．

ウィンドウの状態はウィンドウ管理テーブルに登録してコントロールします．これはウィンドウシステムの状態モデルとなっています．

アプリケーション側からは，このモデルの状態を変える Control 関数群とモデルの状態をスクリーン上に反映させる View 関数群とによってウィンドウシステムが成立します．これを MVC (Model View Controller) モデルといいます．

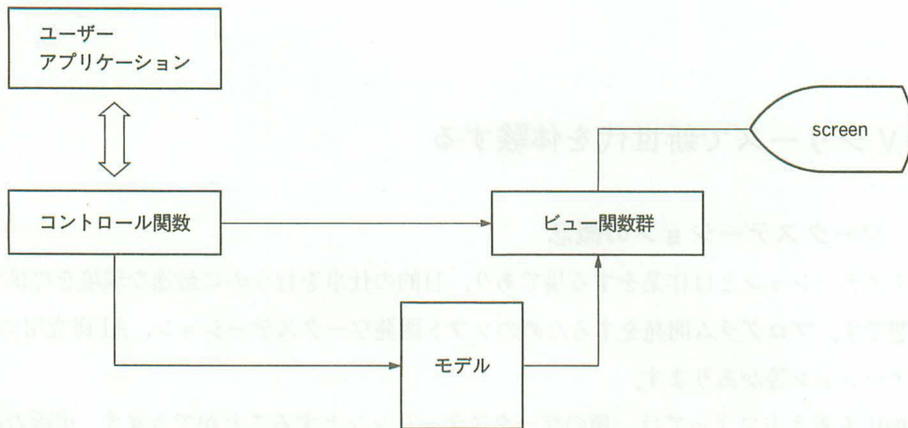


図 5-1-1　MVC モデル

Control 関数群がモデルの状態を変えたときには, View 関数群を呼び出してモデルの新しい状態をスクリーンに反映させます．以下にモデル，Control 関数群，View 関数群を順に説明します．

5-1-2-2　モデル

モデルはウィンドウシステムの状態を管理するテーブルと，その一貫性を保つ関数，およびウィンドウ内に表現するテキストを保つテキストバッファ，フレームバッファによって構成されています．

●管理テーブル

管理テーブルは各ウィンドウオブジェクトの属性を保持します．属性の一覧を次に示します．

・タイトル

・初期表示領域

・オープン／クローズフラグ

・ウィンドウ順位(重ね合わせ等の階層を制御する)

また，オプションとして，

・カラー

・ボーダー幅

・ボーダーカラー

等が指定できるようになっています．

●テキストバッファ

ウィンドウ内に書かれるテキストを格納します．本システムでは，1ウィンドウに対して128バイトのリングバッファを使用します．つまり，128バイトの配列を1つのウィンドウに割り当て，表示開始ポインタと最終位置ポインタを使って，ウィンドウ内に表示される最初の文字のバッファ内の位置とバッファ内の文字列の最終位置を示します．新たにウィンドウに出力する際には，バッファにも文字列が送られ，最終位置ポインタが進みます．ポインタが128を越えると0から始まります．このときに，開始ポインタを越えてしまう場合は，表示開始位置ポインタも進めます．次図に示すように，バッファがリング状になっているようにします．

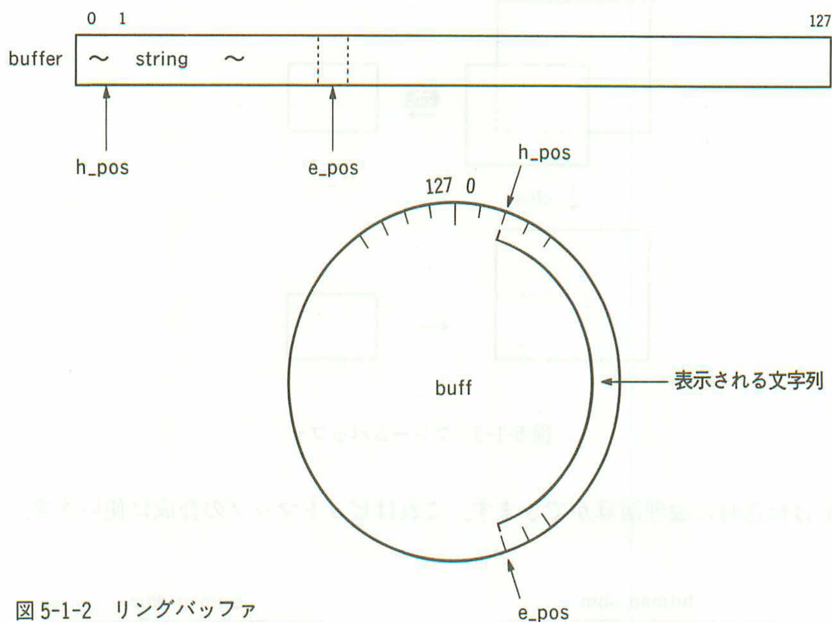


図5-1-2 リングバッファ

●フレームバッファ

本システムでは使用しませんでしたが，一般には必要となる方法にフレームバッファを使用する方法があります．これは，ディスプレイするビットマップをバッファに格納し，ViewとControlでディスプレイバッファへ転送表示する方法です．PC-9800シリーズでは，モノクロ640×400モードだと3〜4プレーンが表示されないプレーンとなるので，それらをフレームバッファとしてウィンドウ内に書きたいグラフィックス等を書きます．実際にスクリーン上に見えるのは，このフレームバッファの1部をBITBLT(ビットブロック転送)でディスプレイバッファ(表示プレーン)に転送されたものです．この際にはビットマップの転送が重要で，この関数のパフォーマンスはウィンドウシステム全体の速度等を決定することがあります．

● BITBLT(BIT Block Transfer)，BYTEBLT(BYTE Block Transfer)

BITBLT(ビットブロック転送)とBYTEBLT(バイトブロック転送)は，ウィンドウシステムを構築する際に重要な役割を果たす関数です．BITBLTは，あるメモリ領域上のビットマップ

を他の領域に転送する関数です．ここに示す関数では，転送はディスプレイバッファと3つのプレーンの間での2方向の転送となっています．BITBLTはウィンドウオーバーラップ，メニュー表示時の退避や再表示に使います．

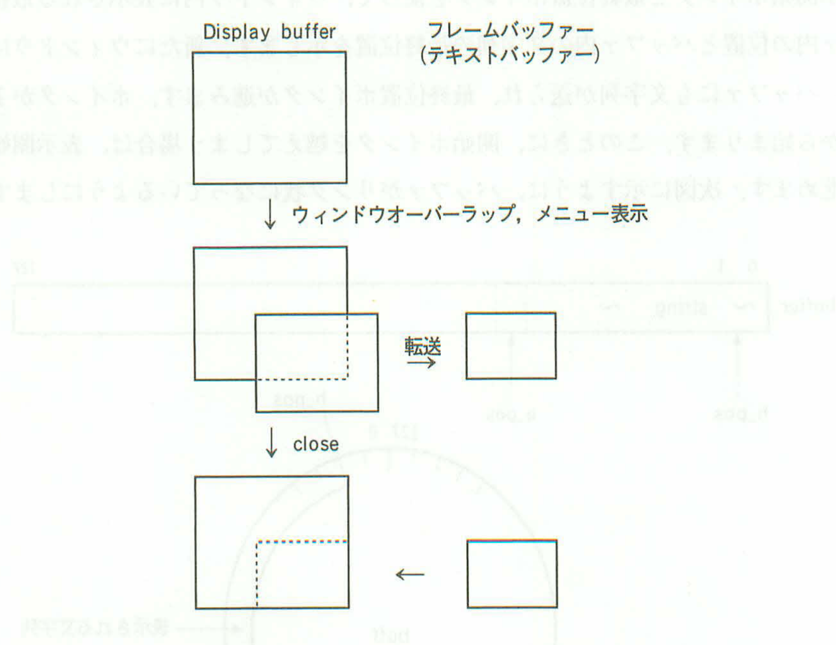


図5-1-3　フレームバッファ

BITBLTは転送時に論理演算ができます．これはビットマップの合成に使います．

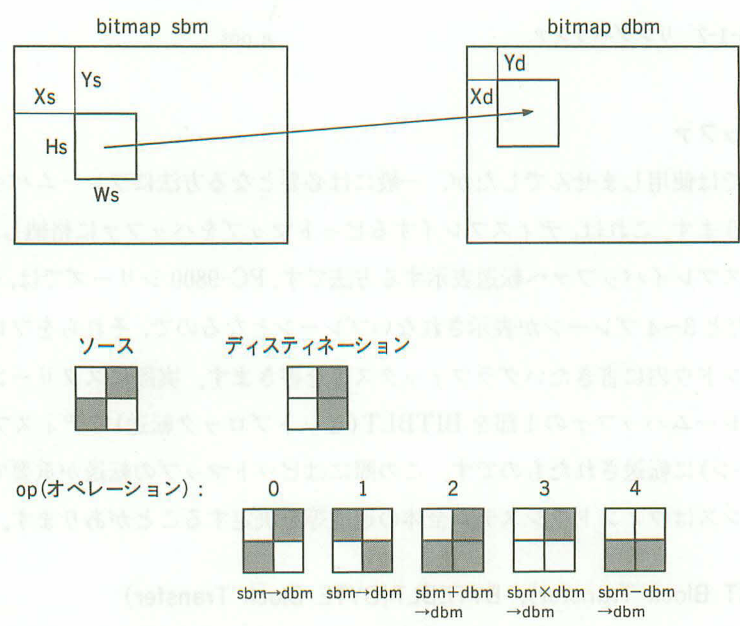


図5-1-4 ビットマップ合成

5-1-2-3 Control

Control 関数群にはモデルに対して種々の演算を施して，表示されるべき状態を作り上げる関数です．さらに，この関数は変化したモデルを表示し，スクリーン上に反映させるために，View 関数群も呼び出します．

Control 関数群はユーザーの必要に合わせていろいろな拡張ができます．基本的なものを以下に示します．

CREATE	：ウィンドウオブジェクトをモデル上に生成する
KILL	：ウィンドウオブジェクトの抹消
OPEN	：ウィンドウを開く
CLOSE	：ウィンドウを閉じる
RESHAPE	：ウィンドウの形を変える
MOVE	：ウィンドウの位置を変える
PUT-STRING	：ウィンドウに文字列を表示する
GET-STRING	：ウィンドウから文字列を読み込む

5-1-2-4 View

View 関数群の役割はモデルの状態をスクリーン上に表示することです．本システムでは Control 関数から呼び出されています．draw_all や buff_to_w などが View 関数群に相当します．

Smalltalk-80 では View は抽象的なクラスとして定義されていますが，ここでは単にモデルを画面上に反映させるファンクションとして考えます．

5-1-2-5 プログラムの仕様

本プログラムは以下に示す仕様を基に設計されています．

▶データ構造

● point：画面上の位置を示す

```
struct  point  {
            int  x;                                    範囲：0～639
            int  y;                                        ：0～399
            };
```

● ch_point：画面上のキャラクタ位置を示す

```
struct  ch_point  {
            int  ch_x;                                 範囲：0～29
            int  ch_y;                                     ：0～24
            };
```

● region：画面上の領域を示す

```
struct region {
        int x;                              範囲：0~639
        int y;                                  ：0~399
        int w;                                  ：0~639
        int h;                                  ：0~399
    };
```

● bitmap：ビットマップ

```
struct bitmap {
        int w;                              範囲：0~639
        int h;                                  ：0~399
    };
```

● window：ウィンドウ

```
struct wmngtbl {
        char    title[10];
        struct  region reg;
        int     crntcx;
        int     crntcy;
        int     opflag;
        int     exist;
    };
```

● c_window：キャラクタ・ウィンドウ

```
struct c_window {
        struct  ch_region  *cw_reg;
        char    *title;
    };
```

● bytemap：バイトマップ

```
struct bytemap {
        int ch_w;
        int ch_h;
    };

struct rectangle {
        int     x1;
        int     y1;
        int     x2;
```

```
        int     y 2;
                  };

struct  wmngtbl  {
        char      title[10];
        struct    region  reg;
        int       crntcx;
        int       crntcy;
        int        opflag;
        int        exist;
                  };

struct  wmngtbl  w_tbl[MAXW];

int  num_of_w;

int  num_of_op_w;

struct  text_buffer  {
        int       h_pos;     /* The first character on the window */
        int       e_pos;     /* The last character on the window */
        char       buffer[128];
                  }  t_buff[MAXW];
```

rectangle：四角形

Text_buffer：テキストバッファー

▶ 主な関数一覧

関数名：create_w

書　式　int create_w (title,region,color,b_color,boarder)

引　数

char *title	；ウィンドウのタイトル
struct region region	；初期オープン領域
int color	；ウィンドウ色 (本書では無視)
int b_color	；ボーダーの色 (〃)
int boarder	；ボーダーのサイズ(〃)

リターン値

ウィンドウが登録された時は 1～MAXWINDOW の間の整数がウインドウ・ディスクリプタとして返されます．ウィンドウが登録できなかった時には 0 が返されます．

機　能

ウィンドウを新たに管理テーブルへ登録します．

関数名：kill_w

書　式　int kill_w (w_id)

引　数　int w_id　；ウィンドウディスクリプタ

機　能

w_id で指定されたウィンドウを管理テーフルより削除します．

関数名：open_w

書　式　open_w (w_id)

引　数　int w_id　；ウィンドウディスクリプタ

リターン値

1：オープン成功

0：オープン失敗

機　能

w_idで指定されたウィンドウをオープンします.

関数名：close_w

書　式　close_w (w_id)

引　数　int　w_id　　；ウィンドウディスクリプタ

機　能

ウィンドウw_idを閉じます.

関数名：move_w

書　式　int　move_w (w_id,new_point)

引　数　int　w_id　　；ウィンドウディスクリプタ

struct　point　new_point　　；新しい場所の左上の点

リターン値

成功：0

失敗：1以上の整数 (エラーナンバー)

機　能

ウィンドウを移動します.

関数名：reshape_w

書　式　int　reshape_w (w_id,new_region)

引　数　int　w_id　　；ウィンドウディスクリプタ

struct　region　new_region　　；新しい形を示す領域

リターン値

成功：0

失敗：1以上 (エラーナンバー)

機　能

ウィンドウの形を変えます．

関数名：put_string

書　式　int　put_string（string,w_id）

引　数　char *string　　；文字列

　　　　int　w_id　　；ウィンドウディスクリプタ

リターン値

成功：0

失敗：1以上の数（エラーナンバー）

機　能

ウィンドウw_idに文字列を出力します．

関数名：put_char

書　式　int　put_char（c,w_id）

引　数　char c　　；文字

　　　　int　w_id　　；ウィンドウディスクリプタ

リターン値

成功：0

失敗：1以上の数（エラーナンバー）

機　能

ウィンドウw_idに文字を出力します．

関数名：get_char

書　式　int　get_char（w_id）

引　数　int　w_id　　；ウィンドウディスクリプタ

リターン値

入力された文字

機能

ウィンドウw_idよりの入力を得ます.

関数名：create_m

書式　int　create_m (title, items, nitems)

引数　char *title　　;メニュータイトル

char *items[]　　;メニューアイテム配列

int　*nitems　　;メニューアイテムの個数

リターン値

メニューidがリターンされます(失敗のときは0がリターン).

機能

引数で定義されたメニューをテーブルへ登録します.

関数名：kill_m

書式　kill_m (m_id)

引数　int　m_id　　;メニューディスクリプタ

機能

メニューm_idをテーブルより削除します.

関数名：open_m

書式　int　open_m (m_id, pt)

引数　int　m_id　　;メニューディスクリプタ

struct　point　pt　　;メニューをオープンする左上座標

リターン値

成功：0

失敗：1以上の数（エラーナンバー）

機能

メニューm_idをptの位置を左上としてオープンします．

関数名：close_m

書式 close_m（m_id）

引数 int m_id ；メニューディスクリプタ

機能

メニュー m_idを閉じます．

関数名：bitblt（ビットブロック転送）

書式 bitblt（sbm,xs,ys,hs,ws,dbm,xd,yd,op）

引数

struct	bitmap	sbm	；ソースビットマップ
int	xs		； x座標 ｝左上コーナー
int	ys		； y座標 ｝左上コーナー
int	hs		； 高さ
int	ws		； 幅
struct	bitmap	dbm	；ディスティネーションビットマップ
int	xd		； x座標 ｝左上コーナー
int	yd		； y座標 ｝左上コーナー
int	op		；オペレーション 0：ノーマル 1：リバース 2：OR 3：AND 4：XOR

機　能

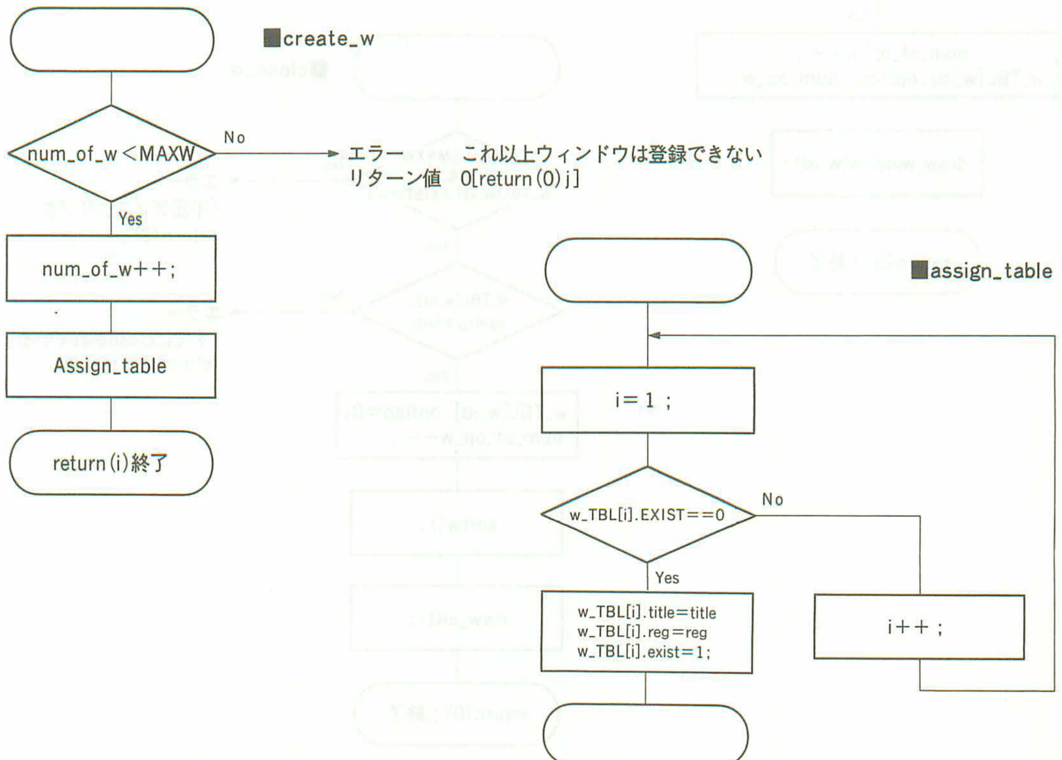
ソースビットマップをopで指定された論理演算を実行してディスティネーションビットマップへ転送します.

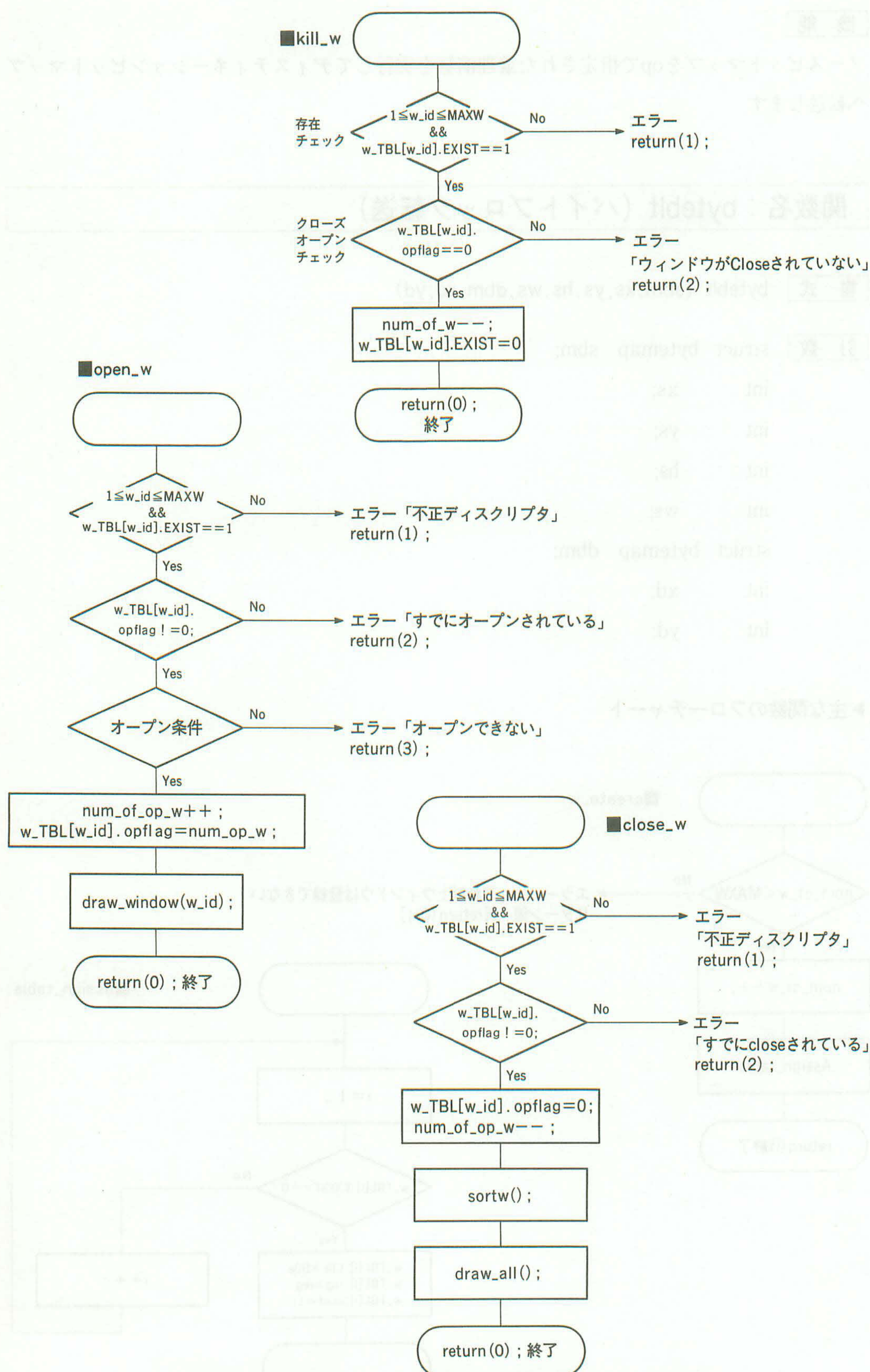
関数名：byteblt（バイトブロック転送）

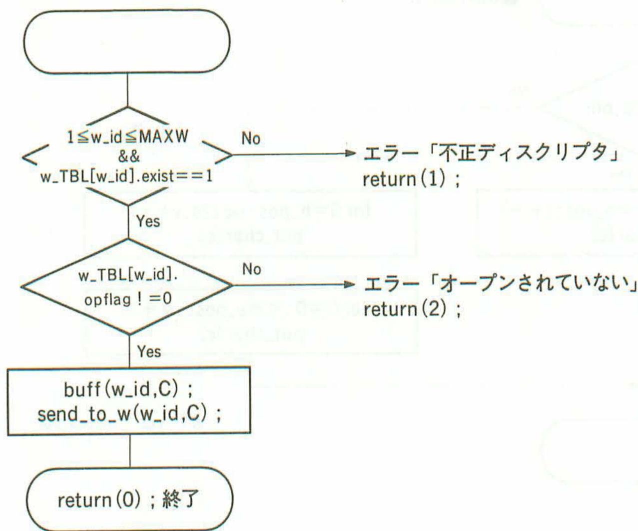
書　式　byteblt（sbm,xs,ys,hs,ws,dbm,xd,yd）

引　数
```
struct  bytemap  sbm;
int         xs;
int         ys;
int         hs;
int         ws;
struct  bytemap  dbm;
int         xd;
int         yd;
```

▶主な関数のフローチャート




■kill_w
存在チェック
1≦w_id≦MAXW && w_TBL[w_id].EXIST==1
No
エラー return(1);
Yes
クローズオープンチェック
w_TBL[w_id]. opflag==0
No
エラー「ウィンドウがCloseされていない」 return(2);
Yes
num_of_w−−; w_TBL[w_id].EXIST=0
return(0); 終了
■open_w
1≦w_id≦MAXW && w_TBL[w_id].EXIST==1
No
エラー「不正ディスクリプタ」 return(1);
Yes
w_TBL[w_id]. opflag ! =0;
No
エラー「すでにオープンされている」 return(2);
Yes
オープン条件
No
エラー「オープンできない」 return(3);
Yes
num_of_op_w++; w_TBL[w_id].opflag=num_op_w;
draw_window(w_id);
return(0); 終了
■close_w
1≦w_id≦MAXW && w_TBL[w_id].EXIST==1
No
エラー「不正ディスクリプタ」 return(1);
Yes
w_TBL[w_id]. opflag ! =0;
No
エラー「すでにcloseされている」 return(2);
Yes
w_TBL[w_id].opflag=0; num_of_op_w−−;
sortw();
draw_all();
return(0); 終了



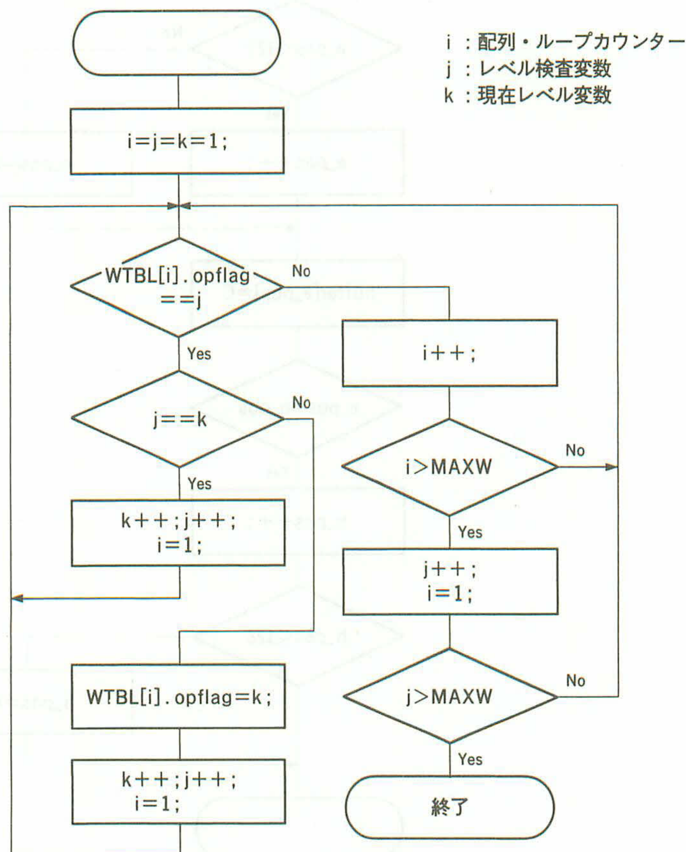
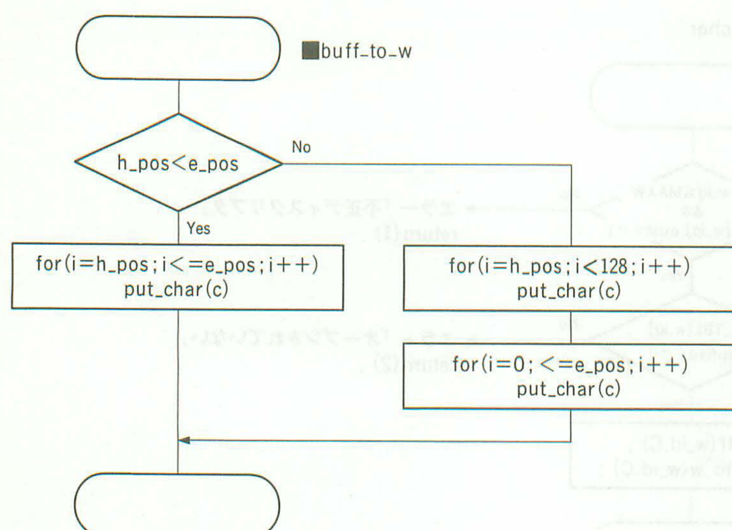
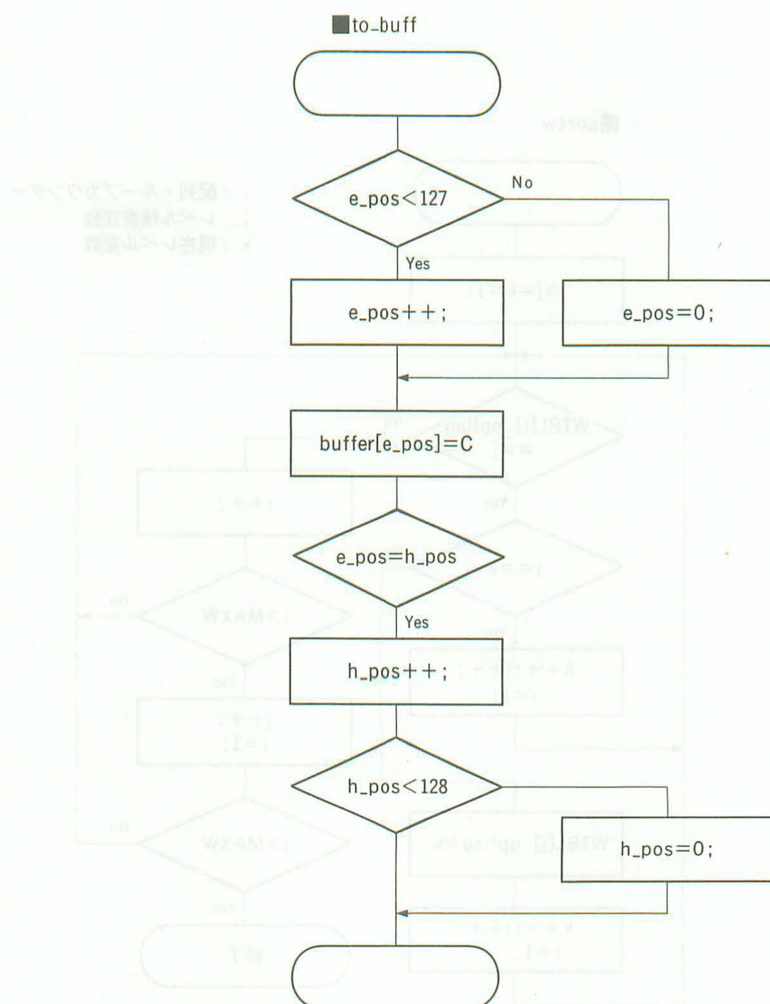


図5-1-5　フローチャート(一部)


■buff_to_w
h_pos<e_pos
No
Yes
for(i=h_pos;i<=e_pos;i++)
put_char(c)
for(i=h_pos;i<128;i++)
put_char(c)
for(i=0;<=e_pos;i++)
put_char(c)


■to_buff
e_pos<127
No
Yes
e_pos++;
e_pos=0;
buffer[e_pos]=C
e_pos=h_pos
Yes
h_pos++;
h_pos<128
h_pos=0;

プログラム 5-1-1　window.c

```
/*window.c
        window maneger library copyright (C) DMSC 1986.10.01
*/

#include "window.h"

/* create_w is a control function to create a window object in the
window manager table. create_w returns window descriptor when succeed.
Zero will be returned when a table is full.
*/

create_w(title,reg,color,b_color,boarder)
char *title;
struct region reg;
int color, b_color, boarder;
{
int i = 1;
        if (num_of_w < MAXW) {
                num_of_w++;
        while ( i < MAXW ) {
                if ( w_tbl[i].exist == 0 ) {
                        strcpy(w_tbl[i].title,title);
                        w_tbl[i].reg = reg;
                        w_tbl[i].exist = 1;
                        return(i);
                        }
                i++;
                }
                }
        else {
                return(0); /* No more objects stack */
                }
}

/* kill_w is a control function to delete a window object from
window table
*/

kill_w(w_id)
{
        if ( 1 <= w_id && w_id <= MAXW && w_tbl[w_id].exist == 1) {
                if ( w_tbl[w_id].opflag == 0 ) {
                        w_tbl[w_id].exist = 0;
                        }
                else {
                        return(2);
                        }
                }
        else {
                return(1);
                }
}

/* open_w is a function to open specified window
*/

open_w(w_id)
int w_id;
{

        if (((1 <= w_id) && (w_id <= MAXW)) && (w_tbl[w_id].exist == 1)) {
```

```
				if(w_tbl[w_id].opflag == 0) {
					if ( opentest(w_id) == 1 ) {
						num_of_op_w++;
						w_tbl[w_id].opflag = num_of_op_w;
						draw_window(w_id);
						return(0);
						}
					else {
						return(3); /* Cannot Open */
						}
					}
				else {
					return(2); /* Already opened */
					}
				}
		else {
				return(1); /* Illigal descriptor */
				}
}

/* opentest checks if specified area is legitimate for the window area */

opentest(w_id)
int w_id;
{
		if(w_tbl[w_id].reg.x > 0 && w_tbl[w_id].reg.x < 80
		  && w_tbl[w_id].reg.y > 0 && w_tbl[w_id].reg.y < 25
		  && (w_tbl[w_id].reg.x + w_tbl[w_id].reg.w < 80)
		  && (w_tbl[w_id].reg.y + w_tbl[w_id].reg.h < 25)) {
				return(1);
				}
		else return(0);
}

/* close_w closes a specified window */

close_w(w_id)
int w_id;
{
		if (((1<= w_id) && (w_id <= MAXW)) && (w_tbl[w_id].exist == 1)) {
				if (w_tbl[w_id].opflag != 0) {
						w_tbl[w_id].opflag = 0;
						num_of_op_w--;
						sortw();
						draw_all();
						return(0);
						}
				else {
						 return(2);
						}
				}
		else {
				return(1);
				}
}

/* sortw reorder window hierachy when any modification is made */

sortw()
{
int i, j, k;
i = j = k = 1;
		loop:
		if ( w_tbl[i].opflag == j ) {
```

```
                if ( j == k) {
                        k++;
                        j++;
                        i = 1;
                        goto loop;
                        }
                else {
                        w_tbl[i].opflag = k;
                        k++;
                        j++;
                        i = 1;
                        goto loop;
                        }
                }
        else {
                i++;
                if ( i > MAXW ) {
                        j++;
                        i = 1;
                        if ( j > MAXW ) return(0);
                        else goto loop;
                        }
                else goto loop;
                }
}

/* draw_all is a view function to redisplay every window opening */

draw_all()
{
int i, j;
        t_cls();
        g_line(0,0,639,399,7,2,0,0);
        i = 1;
        while(i <= MAXW) {

                for(j = 0; j <  MAXW; j++) {

                        if (w_tbl[j].opflag == i) {
                                draw_window(j);
                                }
                }
        i++;
        }
}

/* draw_window is a view function to draw window frame, title and text */

draw_window(w_id)
int w_id;
{
struct rectangle rect;

        g_line(w_tbl[w_id].reg.x * 8,
             w_tbl[w_id].reg.y * 16,
            (w_tbl[w_id].reg.w + w_tbl[w_id].reg.x) * 8,
            (w_tbl[w_id].reg.h + w_tbl[w_id].reg.y) * 16,
                4,2,0);
        g_line(w_tbl[w_id].reg.x * 8,
             w_tbl[w_id].reg.y * 16,
            (w_tbl[w_id].reg.w + w_tbl[w_id].reg.x) * 8,
            (w_tbl[w_id].reg.h + w_tbl[w_id].reg.y) * 16,
                0,1,0);
        rect.x1 =  w_tbl[w_id].reg.x  * 8 ;
```

```
        rect.y1 =  ( w_tbl[w_id].reg.y - 1 ) * 16 ;
        rect.x2 =  ( w_tbl[w_id].reg.x + w_tbl[w_id].reg.w ) * 8;
        rect.y2 =  w_tbl[w_id].reg.y * 16;
        r_box(rect);
        cls_t(w_id);
        w_tbl[w_id].crntcx = 0;
        w_tbl[w_id].crntcy = 0;
        dsp_title(w_id);
        buff_to_w(w_id);
}

/* dsp_title prints title into window label area */

dsp_title(w_id)
int w_id;
{
int i;
        t_loc(w_tbl[w_id].reg.x,w_tbl[w_id].reg.y - 1);
        for(i = 0; i < w_tbl[w_id].reg.w && w_tbl[w_id].title[i] != '¥0';i++) {
                putchar(w_tbl[w_id].title[i]);
                }
}

/* put_char sends a character to the window and to the buffer */

put_char(c,w_id)
char c;
int w_id;
{
        if ( 1 <= w_id && w_id <= MAXW && w_tbl[w_id].exist == 1 ) {
                if ( w_tbl[w_id].opflag != 0 ){
                        to_buff(w_id,c);
                        send_to_w(w_id,c);
                        return(0);
                        }
                else {
                        return(2);
                        }
                }
        else {
                return(1);
                }
}

/* get_char returns a character from the window stream */

get_char(w_id)
int w_id;
{
        if ( 1 <= w_id && w_id <= MAXW && w_tbl[w_id].exist == 1 ) {
                if ( w_tbl[w_id].opflag != 0 ){
                        to_buff(w_id,from_w(w_id));
                        return(0);
                        }
                else {
                        return(2);
                        }
                }
        else {
                return(1);
                }
}

/* to_buff sends character to text ring buffer */
```

```
to_buff(w_id.c)
int w_id;
char c;
{
int i, j;
        i =  t_buff[w_id].e_pos;
        j =  t_buff[w_id].h_pos;
        if ( i < 127 ) {
                        i++;
                        }
                else {
                        i = 0;
                        }
        t_buff[w_id].buffer[i] = c;
        if ( i == j ) {
                j++;
                }
        if (!( j < 128 )) {
                j = 0;
                }
        t_buff[w_id].e_pos = i;
        t_buff[w_id].h_pos = j;
}

/* buff_to_w sends ring buffer content to the display buffer */

buff_to_w(w_id)
int w_id;
{
int i, e, h;
        e = t_buff[w_id].e_pos;
        h = t_buff[w_id].h_pos;

        if ( e == h ) { return(0); }
        if ( h < e ) {
                for ( i = h ; i <= e ; i++ ) {
                        send_to_w(w_id,t_buff[w_id].buffer[i]);
                        }
                }
        else {
                for ( i = h ; i < 128 ; i++ ) {
                        send_to_w(w_id,t_buff[w_id].buffer[i]);
                        }
                for ( i = 0 ; i <= e ; i++ ) {
                        send_to_w(w_id,t_buff[w_id].buffer[i]);
                        }
                }
}

/* send_to_w sends a character to the window */

send_to_w(w_id,c)
int w_id;
char c;
{
int h, e, d;
        h = t_buff[w_id].h_pos;
        e = t_buff[w_id].e_pos;

        if (!(w_tbl[w_id].crntcx < w_tbl[w_id].reg.w )){
                w_tbl[w_id].crntcx = 0;
                w_tbl[w_id].crntcy++;
                }
```

```
        if (!( w_tbl[w_id].crntcy < w_tbl[w_id].reg.h)){
                if ( h + w_tbl[w_id].reg.w > 127 ) {
                        h = ( w_tbl[w_id].reg.w - ( 127 - h ));
                        }
                else {
                        h = h + w_tbl[w_id].reg.w;
                        }
                t_buff[w_id].h_pos = h;
                w_tbl[w_id].crntcy = 0;
                w_tbl[w_id].crntcx = 0;
                cls_t(w_id);
                buff_to_w(w_id);

                if ( w_tbl[w_id].crntcy >= w_tbl[w_id].reg.h ) {
                        w_tbl[w_id].crntcy = w_tbl[w_id].reg.h;
                        }
                }
                t_loc(w_tbl[w_id].crntcx + w_tbl[w_id].reg.x,
                      w_tbl[w_id].crntcy + w_tbl[w_id].reg.y);
                putchar(c);
                w_tbl[w_id].crntcx++;
}

/* from_w gets a character from a specified window */

from_w(w_id)
int w_id;
{
int h, e, d;
char c;
        h = t_buff[w_id].h_pos;
        e = t_buff[w_id].e_pos;

        c = getchi();
        putchar(c);
        w_tbl[w_id].crntcx++;
        if (!(w_tbl[w_id].crntcx < w_tbl[w_id].reg.w )){
                w_tbl[w_id].crntcx = 0;
                w_tbl[w_id].crntcy++;
                }
        if (!( w_tbl[w_id].crntcy < w_tbl[w_id].reg.h)){
                if ( h + w_tbl[w_id].reg.w > 127 ) {
                        h = ( w_tbl[w_id].reg.w - ( 127 - h ));
                        }
                else {
                        h = h + w_tbl[w_id].reg.w;
                        }
                t_buff[w_id].h_pos = h;
                w_tbl[w_id].crntcy = 0;
                w_tbl[w_id].crntcx = 0;
                cls_t(w_id);
                buff_to_w(w_id);

                if ( w_tbl[w_id].crntcy >= w_tbl[w_id].reg.h ) {
                        w_tbl[w_id].crntcy = w_tbl[w_id].reg.h;
                        }
                }
                t_loc(w_tbl[w_id].crntcx + w_tbl[w_id].reg.x,
                      w_tbl[w_id].crntcy + w_tbl[w_id].reg.y);

        return(c);
}
```

```
put_string(string,w_id)
char *string;
int w_id;
{
        while( *string != '¥0' ) {
                put_char(*string,w_id);
                string++;
                }
}


struct region *m_get_reg()
{
struct region *reg, *buff;
int lbtn, rbtn;
int xa, ya, xb, yb;
int abs_x, abs_y;
lbtn = 0;
        m_csron();
        while ( lbtn == 0 ) {
                m_pos(&abs_x,&abs_y,&lbtn,&rbtn);
                }
        xa = abs_x; ya = abs_y;
        while ( lbtn != 0 ) {
                m_pos(&abs_x,&abs_y,&lbtn,&rbtn);
        if ( ya < abs_y ) {
                reg->h =  abs_y - ya ;
                reg->y = ya ;
                }
        else {
                reg->h =  ya - abs_y ;
                reg->y = abs_y ;
                }
        if ( xa < abs_x ) {
                reg->w =  abs_x - xa ;
                reg->x = xa ;
                }
        else {
                reg->w =  xa - abs_x ;
                reg->x = abs_x ;
                }

                g_boxx(buff);
                g_boxx(reg);
                buff = reg;
                }

                reg->h = reg->h / 16 ;
                reg->y = reg->y / 16 ;
                reg->w = reg->w / 8 ;
                reg->x = reg->x / 8 ;

        m_csroff();
        return(reg);
}

init_w()
{
int i;

        g_init();
        m_init();
        g_screen(3,0,0,1);
```

```
        t_cls();
        g_cls();
        t_color(0x41);
        g_line(0,0,639,399,7,2,0,0);
        for ( i = 0; i < MAXW ; i++) {
                t_buff[i].e_pos = 127;
                t_buff[i].h_pos = 127;
                t_buff[i].buffer[0] = '¥0';
                }
}

r_box(rect)
struct rectangle rect;
{
        g_line(rect.x1,rect.y1,rect.x2,rect.y2,6,2,0,0,0);
        g_line(rect.x1,rect.y1,rect.x2,rect.y2,0,1,0,0,0);
}

reshape_w(w_id)
int w_id;
{
struct region reg;
        reg = *m_get_reg();
        close_w(w_id);
        w_tbl[w_id].reg = reg;
        open_w(w_id);
}

move_w(w_id)
int w_id;
{
struct region *reg, *buff;
int lbtn, rbtn, x, y;
lbtn = 0;
        reg->h = w_tbl[w_id].reg.h * 16;
        reg->w = w_tbl[w_id].reg.w * 8;

        m_csron();
        while( lbtn == 0 ) {
                m_pos(&x,&y,&lbtn,&rbtn);
                }
        while( lbtn != 0 ) {
                m_pos(&x,&y,&lbtn,&rbtn);
                reg->x = x;
                reg->y = y;
                g_boxx(buff);
                g_boxx(reg);
                buff = reg;
                }
        m_csroff();
        close_w(w_id);
        w_tbl[w_id].reg.x = reg->x / 8 ;
        w_tbl[w_id].reg.y = reg->y / 16;
        open_w(w_id);
}

cls_t(w_id)
int w_id;
{
int i, j, x1, x2, y1, y2;
        x1 = w_tbl[w_id].reg.x;
        x2 = w_tbl[w_id].reg.w + x1;
        y1 = w_tbl[w_id].reg.y;
```

```
        y2 = w_tbl[w_id].reg.h + y1;
        for ( j = y1 ; j < y2 ; j++ ) {
                for ( i = ( x1 - 1 ) ; i < x2 ; i++ ) {
                        t_loc(i,j);
                        putchar(' ');
                        }
        }
}

g_boxx(reg)
struct region *reg;
{
        g_linex(reg->x,reg->y,reg->x,reg->y + reg->h,1);
        g_linex(reg->x,reg->y,reg->x + reg->w,reg->y,1);
        g_linex(reg->x + reg->w,reg->y,reg->x + reg->w,reg->y + reg->h,1);
        g_linex(reg->x,reg->y + reg->h,reg->x + reg->w,reg->y + reg->h,1);
}

boxx(x1,y1,x2,y2)
int x1, y1, x2, y2;
{
        g_linex(x1,y1,x1,y2,1);
        g_linex(x1,y1,x2,y1,1);
        g_linex(x1,y2,x2,y2,1);
        g_linex(x2,y1,x2,y2,1);
}
```

プログラム5-1-2　window.h

```
/* window.h
        window maneger library header file copyright (C) DMSC 1986.10.01
*/

#define MAXW 4


struct region {
        int x;
        int y;
        int w;
        int h;
        };

struct rectangle {
        int x1;
        int y1;
        int x2;
        int y2;
        };


struct wmngtbl {
        char title[10];
        struct region reg;
        int crntcx;
        int crntcy;
        int  opflag;
        int  exist;
        };
```

```
struct wmngtbl w_tbl[MAXW];

int num_of_w;

int num_of_op_w;

struct text_buffer {
        int h_pos;      /* The first character on the window */
        int e_pos;      /* The last character on the window */
        char buffer[128];
        } t_buff[MAXW];
```

プログラム5-1-3 wcat.c

```
/*wcat.c
        window cat copyright (C) DMSC 1986.10.01
*/

#include "stdio.h"
#include "window.c"

main()
{
int w1, w2, w3, abs_x, abs_y, lbtn, rbtn, i;
struct region *r1, *r2, *r3, *m_get_reg();
char b1[80], c, ch, s[10];
        FILE *fd, *fopen();

lbtn = 0;

        init_w();

        r1->x = 50; r1->y = 5; r1->w = 15; r1->h = 10;
        w1 = create_w("Select",*r1,1,1,1);
        open_w(w1);
loop:
        put_string("1:Show file    ",w1);
        put_string("Else quit      ",w1);
        c = from_w(w1);

if ( c == '1' ) {
        i = 0;
        r2 = m_get_reg();
        w2 = create_w("Input",*r2,1,1,1);
        open_w(w2);
        put_string("File name:",w2);
        while( (( ch = from_w(w2) ) != 13 )
                && ( i < 10 )) {
                s[i] = ch;
                i++;
                }
                s[i] = '¥0';
        if (( fd = fopen(s,"r")) == NULL) {
                put_string("Can't open file ¥n",w2);
                }
        else {
                r3 = m_get_reg();
                w3 = create_w(s,*r3,1,1,1);
                open_w(w3);
```

```
            while ( fgets(bl,80,fd) != 0 ) {
                    put_string(bl,w3);
            }
            fclose(fd);
            }
        }

}
```

プログラム5-1-4 Bitblt

```
; Bitblt Screen Windows  for Lattice C Version
;
;    Copyright (C) Fukuma Ymazaki
;
;******************************************************************************
; マクロ定義
;******************************************************************************

push_reg    macro  r                    ; Regster push Macro
      irp    reg,<r>
      push   reg
      endm
endm

pop_reg             macro  r            ; Regster pop Macro
      irp    reg,<r>
      pop    reg
      endm
endm

gvrammacro page                         ; Gvram Page Select Macro
      push   ax
      mov    al,page
      out    0a6h,al
      pop    ax
      endm

printmacro buffers                      ; Function No 9
      push   ds
      push   ax
      push   dx
      push   cs
      pop    ds
      mov    dx,offset buffers
      mov    ah,9
      int    21h
      pop    dx
      pop    ax
      pop    ds
      endm

;******************************************************************************
;                   ※※ デバッグ・スイッチ使用法 ※※
;******************************************************************************
;
; デバッグ・スイッチ(debug)をonにすると，パラメータが範囲を
; 越えた場合，エラーメッセージを出力する．
;
;******************************************************************************
; 既定値定義
;******************************************************************************
```

```
on          = 1             ; Switch on
off         = 0             ; Switch off
debug       = on            ; Debug Message Switch (on or off)
disp_seg    = 0a800h        ; Display Addres
page_1_seg  = 0a800h        ; Buffer Page 1
page_2_seg  = 0afd0h        ; Buffer Page 2
page_3_seg  = 0b7a0h        ; Buffer Page 3

;**********************************************************************
; Lattice C インプリメント領域
;**********************************************************************
        include DOS.MAC

        if      LPROG
offs =          6
        else
offs =          4
        endif

        pseg

        public BITBLT

        if      LPROG
BITBLT          proc    far
        else
BITBLT          proc    near
        endif

;**********************************************************************
; bitblt関数のパラメータ
;**********************************************************************
;
;       int xs,ys,hs,ws,xd,yd,op,pg,dr ;
;
;       bitblt ( xs,ys,hs,ws,xd,yd,op,pg,dr )
;
;       xs      =       Source X Position       (Word)
;       ys      =       Source Y Position       (Word)
;       ws      =       Data Width              (Word)
;       hs      =       Data Hight              (Word)
;       xd      =       Destination X Position          (Word)
;       yd      =       Destination Y Position          (Word)
;       op      =       Operation Flag                  (Byte)
;       pg      =       Page Serect             (Byte)
;       dr      =       Direction Flag                  (Byte)
;
;**********************************************************************
; Main Routin
;**********************************************************************

        push    bp
        mov     bp,sp
        push_reg        <ax,bx,cx,dx,si,di,es,ds>
if      LDATA
        mov     ax,offs[bp]
        mov     ds,ax
        mov     ax,(offs    )[bp]
        mov     word ptr cs:XS,ax
        mov     ax,(offs+ 2)[bp]
        mov     word ptr cs:YS,ax
        mov     ax,(offs+ 4)[bp]
```

```
     mov    cs:HS,ax
     mov    ax,(offs+ 6)[bp]
     mov    cs:WS,ax
     mov    ax,(offs+ 8)[bp]
     mov    cs:XD,ax
     mov    ax,(offs+10)[bp]
     mov    cs:YD,ax
     mov    ax,(offs+12)[bp]
     mov    cs:OP,al
     mov    ax,(offs+14)[bp]
     mov    cs:PG,al
     mov    ax,(offs+16)[bp]
     mov    cs:DR,al
else
     mov    ax,ds
     mov    es,ax
     mov    ax,(offs    )[bp]
     mov    cs:XS,ax
     mov    ax,(offs+ 2)[bp]
     mov    cs:YS,ax
     mov    ax,(offs+ 4)[bp]
     mov    cs:HS,ax
     mov    ax,(offs+ 6)[bp]
     mov    cs:WS,ax
     mov    ax,(offs+ 8)[bp]
     mov    cs:XD,ax
     mov    ax,(offs+10)[bp]
     mov    cs:YD,ax
     mov    ax,(offs+12)[bp]
     mov    cs:OP,al
     mov    ax,(offs+14)[bp]
     mov    cs:PG,al
     mov    ax,(offs+16)[bp]
     mov    cs:DR,al
endif

     call   parameter_chack
     cmp    cs:error_flag,1
     je     no_no
     call   _bitblt

no_no:      call    mem_clear
     pop_reg            <ds,es,di,si,dx,cx,bx,ax>
     mov    sp,bp
     pop    bp
     ret           ; End of Main Routin
BITBLT      endp

;*************************************************************************
;    DS = ソース セグメント
;    ES = デスティネーション セグメント
;    SI = ソース オフセット
;    DI = デスティネーション オフセット
;*************************************************************************
;    XS = X座標開始点ソース側（文字数計算  0－79）
;    YS = Y座標開始点ソース側（文字数計算  0－24）
;    HS = Y方向の幅         （文字数計算  1－25）
;    WS = X方向の幅         （文字数計算  1－80）
;    XD = X座標開始点デスティネーション側（文字数計算  0－79）
;    YD = Y座標開始点デスティネーション側（文字数計算  0－24）
;    OP = 論理演算オペレーション    ( 0 - 4 )
;                              0 = ﾉｰﾏﾙ
;                              1 = ﾘﾊﾞｰｽ
;                              2 = or
```

```
;                                      3 = and
;                                      4 = xor
;    PG = バッファ ページング ( 1 - 4 )
;    DR = ディレクション     ( 0 or 1 )
;                                  0 = 表示領域からバッファへ
;                                  1 = バッファから表示領域へ
;*****************************************************************

_bitblt     proc    near
        push_reg <ax,bx,cx,si,di,ds,es>   ; 使用レジスタ退避
        ;------------------------------------------------------------
        cmp     byte ptr cs:DR,0
        je      crt_to_buffers
        cmp     byte ptr cs:DR,1
        je      buffers_to_crt
        ;------------------------------------------------------------
crt_to_buffers:                                 ; 表示領域からバッファーへ
        cmp     cs:PG,1
        jne     next_2
        mov     ax,page_1_seg
        mov     es,ax
        jmp     job_2
next_2:     cmp     cs:PG,2
        jne     next_3
        mov     ax,page_2_seg
        mov     es,ax
        jmp     job_2
next_3:     cmp     cs:PG,3
        jne     next_4
        mov     ax,page_3_seg
        mov     es,ax
        jmp     job_2
next_4:     cmp     cs:PG,4
        jne     job_2
        mov     ax,cs
        mov     es,ax
        jmp     job_2
job_2:      mov     ax,disp_seg
        mov     ds,ax
        jmp     set_up
        ;------------------------------------------------------------
buffers_to_crt:                                 ; バッファーから表示領域へ
        cmp     cs:PG,1
        jne     next_5
        mov     ax,page_1_seg
        mov     ds,ax
        jmp     job_3
next_5:     cmp     cs:PG,2
        jne     next_6
        mov     ax,page_2_seg
        mov     ds,ax
        jmp     job_3
next_6:     cmp     cs:PG,3
        jne     next_7
        mov     ax,page_3_seg
        mov     ds,ax
        jmp     job_3
next_7:     cmp     cs:PG,4
        je      next_8
        jmp     job_3
next_8:     mov     ax,cs
        mov     ds,ax
job_3:      mov     ax,disp_seg
```

```
        mov     es,ax
        ;-------------------------------------------------------------
set_up:         mov     ax,1280                 ; SI = ( 80 * 16 ) * YS ) + XS
        mov     bx,cs:YS
        mul     bx
        add     ax,cs:XS
        mov     si,ax
        cmp     byte ptr cs:PG,4
        jne     next_9
        cmp     byte ptr cs:DR,1
        jne     next_9
        mov     bx,offset PAGE_BUFFERS
        add     si,bx
next_9:
        ;-------------------------------------------------------------
        mov     ax,1280                 ; DI = ( 80 * 16 ) * YD ) + XD
        mov     bx,cs:YD
        mul     bx
        add     ax,cs:XD
        mov     di,ax
        cmp     byte ptr cs:PG,4
        jne     next10
        cmp     byte ptr cs:DR,0
        jne     next10
        mov     bx,offset PAGE_BUFFERS
        add     di,bx
next10:
        ;-------------------------------------------------------------
        mov     ax,cs:HS                ; HS = HS * 16
        mov     bx,16
        mul     bx
        mov     cs:HS,ax
        ;-------------------------------------------------------------
        cmp     byte ptr cs:PG,4        ; if page 4 then page_4_job
        jne     next11
        jmp     page_4_job

next11: cmp byte ptr cs:DR,1            ; For Crt to buffers
        je      crt_out_type_2

crt_out_type_1:
        gvram   0
        mov     al,ds:[si]
        call    operation
        gvram   1
        mov     es:[di],al
        inc     si
        inc     di
        inc     cs:XCOUNT
        mov     bx,cs:WS
        cmp     cs:XCOUNT,bx
        jne     crt_out_type_1
        mov     bx,80
        sub     bx,cs:WS
        add     si,bx
        add     di,bx
        mov     cs:XCOUNT,0
        inc     cs:YCOUNT
        mov     bx,cs:HS
        cmp     cs:YCOUNT,bx
        jne     crt_out_type_1
        jmp     return_job
        ;-------------------------------------------------------------
```

```
crt_out_type_2:
        gvram   1
        mov     al,ds:[si]
        call    operation
        gvram   0
        mov     es:[di],al
        inc     si
        inc     di
        inc     cs:XCOUNT
        mov     bx,cs:WS
        cmp     cs:XCOUNT,bx
        jne     crt_out_type_2
        mov     bx,80
        sub     bx,cs:WS
        add     si,bx
        add     di,bx
        mov     cs:XCOUNT,0
        inc     cs:YCOUNT
        mov     bx,cs:HS
        cmp     cs:YCOUNT,bx
        jne     crt_out_type_2
        jmp     return_job
        ;-------------------------------------------------------------
page_4_job:
        mov     al,ds:[si]
        call    operation
        mov     es:[di],al
        inc     si
        inc     di
        inc     cs:XCOUNT
        mov     bx,cs:WS
        cmp     cs:XCOUNT,bx
        jne     page_4_job
        mov     bx,80
        sub     bx,cs:WS
        add     si,bx
        add     di,bx
        mov     cs:XCOUNT,0
        inc     cs:YCOUNT
        mov     bx,cs:HS
        cmp     cs:YCOUNT,bx
        jne     page_4_job
        ;-------------------------------------------------------------
return_job:
        call    mem_clear
        pop_reg         <es,ds,di,si,cx,bx,ax>
        ret

_bitblt         endp
        ;-------------------------------------------------------------
operation       proc    near    ; 論理演算
        cmp     cs:OP,0                 ; IF OP = 0 THEN GOTO pset
        je      pset
        cmp     cs:OP,1                 ; IF OP = 1 THEN GOTO preset
        je      preset
        cmp     cs:OP,2                 ; IF OP = 2 THEN GOTO p_or
        je      p_or
        cmp     cs:OP,3                 ; IF OP = 3 THEN GOTO p_and
        je      p_and
        cmp     cs:OP,4                 ; IF OP = 4 THEN GOTO p_xor
        je      p_xor
        jmp     pset            ; オペレーションが既定値以外ならノーマル出力
pset:ret                        ; そのまま出力
```

```
preset:     mov    al,0          ; Nulを出力
    ret
p_or:mov    ah,es:[di]
    or     al,ah         ; 転送先のデータとorし，結果を出力
    ret
p_and:      mov    ah,es:[di]
    and    al,ah         ; 転送先のデータとandし，結果を出力
    ret
p_xor:      mov    ah,es:[di]
    xor    al,ah         ; 転送先のデータとxorし，結果を出力
    ret
operation   endp
    ;-----------------------------------------------------------------
mem_clear   proc   near  ; Buffer Memory Clear
    mov    ax,cs
    mov    es,ax
    mov    di,offset XS
    mov    cx,21
    mov    ax,0
    cld
    rep    stosb
    gvram  0             ; Select Gvram Page 0
    ret
mem_clear   endp

;************************************************************************
; パラメータ・チェック
;************************************************************************

parameter_chack   proc   near
    cmp    cs:xs,80
    jnb    error_1
    jmp    par_2
error_1:
    jmp    exit_1
par_2:
    cmp    cs:ys,25
    jnb    error_2
    jmp    par_3
error_2:
    jmp    exit_2
par_3:
    cmp    cs:hs,26
    jnb    error_3
    jmp    par_4
error_3:
    jmp    exit_3
par_4:
    cmp    cs:ws,81
    jnb    error_4
    jmp    par_5
error_4:
    jmp    exit_4
par_5:
    cmp    cs:xd,80
    jnb    error_5
    jmp    par_6
error_5:
    jmp    exit_5
par_6:
    cmp    cs:yd,25
    jnb    error_6
    jmp    par_7
```

```
error_6:
        jmp     exit_6
par_7:
        cmp     cs:op,5
        jnb     error_7
        jmp     par_8
error_7:
        jmp     exit_7
par_8:
        cmp     cs:pg,4
        jnb     error_8
        jmp     par_9
error_8:
        jmp     exit_8
par_9:
        cmp     cs:dr,2
        jnb     error_9
        ret
error_9:
        jmp     exit_9

exit_1: if      debug
        print   perror_1
        endif
        jmp     exit_0
exit_2: if      debug
        print   perror_2
        endif
        jmp     exit_0
exit_3: if      debug
        print   perror_3
        endif
        jmp     exit_0
exit_4: if      debug
        print   perror_4
        endif
        jmp     exit_0
exit_5: if      debug
        print   perror_5
        endif
        jmp     exit_0
exit_6: if      debug
        print   perror_6
        endif
        jmp     exit_0
exit_7: if      debug
        print   perror_7
        endif
        jmp     exit_0
exit_8: if      debug
        print   perror_8
        endif
        jmp     exit_0
exit_9: if      debug
        print   perror_9
        endif

exit_0: if      debug
        print   keywait
        mov     ah,1
        int     21h
        endif
        mov     cs:ERROR_FLAG,1
```

```
     mov     dl,7
     mov     ah,2
     int     21h
     ret
parameter_chack    endp

;*********************************************************************
; バッファー エリア
;*********************************************************************

     XS            dW     ?      ; Source X Position
     YS            dW     ?      ; Source Y Position
     WS            dW     ?      ; Data Width
     HS            dW     ?      ; Data Hight
     XD            dw     ?      ; Destination X Position
     YD            dw     ?      ; Destination Y Position
     OP            db     ?      ; Operation Flag
     PG            db     ?      ; Page Serect
     DR            db     ?      ; Direction Flag
     XCOUNT        dw     ?      ; Direction X Counter
     YCOUNT        dw     ?      ; Direction Y Counter
     ERROR_FLAG    db     ?
     PAGE_BUFFERS  db     32000 dup(?) ; For Page 4 Buffers

;*********************************************************************
; メッセージ エリア
;*********************************************************************

     if debug

perror_1     db     13,10,'xsパラメータが範囲を越えてます.',13,10,'$'
perror_2     db     13,10,'ysパラメータが範囲を越えてます.',13,10,'$'
perror_3     db     13,10,'hsパラメータが範囲を越えてます.',13,10,'$'
perror_4     db     13,10,'wsパラメータが範囲を越えてます.',13,10,'$'
perror_5     db     13,10,'xdパラメータが範囲を越えてます.',13,10,'$'
perror_6     db     13,10,'ydパラメータが範囲を越えてます.',13,10,'$'
perror_7     db     13,10,'opパラメータが範囲を越えてます.',13,10,'$'
perror_8     db     13,10,'pgパラメータが範囲を越えてます.',13,10,'$'
perror_9     db     13,10,'drパラメータが範囲を越えてます.',13,10,'$'
keywait             db     13,10,'任意のキーを押して下さい.',13,10,'$'

     endif

     endps
     end
```

プログラム5-1-5 Byeblt

```
; Byteblt Screen Windows      for Lattice C Version
;
;     Copyright (C) Fukuma Ymazaki
;
;*********************************************************************
; マ ク ロ 定 義
;*********************************************************************

push_reg       macro    r                    ; Regster push Macro
     irp       reg,<r>
     push      reg
     endm
endm
```

```
pop_reg                 macro   r                   ; Regster pop Macro
        irp     reg,<r>
        pop     reg
        endm
endm

gvram macro page                          ; Gvram Page Select Macro
        push    ax
        mov     al,page
        out     0a6h,al
        pop     ax
        endm

print macro     buffers                   ; Function No 9
        push    ds
        push    ax
        push    dx
        push    cs
        pop     ds
        mov     dx,offset buffers
        mov     ah,9
        int     21h
        pop     dx
        pop     ax
        pop     ds
        endm
;*********************************************************************
; 既 定 値 定 義
;*********************************************************************

on              = 1             ; Switch on
off             = 0             ; Switch off
debug           = off           ; Debug Message Switch (on or off)
disp_seg        = 0a000h        ; Display Addres
attb_seg        = 0a200h        ; Display Attribute Addres

;*********************************************************************
; L a t t i c e  C  インプリメント領域
;*********************************************************************

        include DOS.MAC

        if      LPROG
offs    =       6
        else
offs    =       4
        endif

        pseg

        public  BYTEBLT

        if      LPROG
BYTEBLT         proc    far
        else
BYTEBLT         proc    near
        endif
;*********************************************************************
; M a i n   R o u t i n
;*********************************************************************

        push    bp
        mov     bp,sp
```

```
        push_reg        <ax,bx,cx,dx,si,di,es,ds>
if      LDATA
        mov     ax,offs[bp]
        mov     ds,ax
        mov     ax,(offs    )[bp]
        mov     word ptr cs:XS,ax
        mov     ax,(offs+ 2)[bp]
        mov     word ptr cs:YS,ax
        mov     ax,(offs+ 4)[bp]
        mov     cs:HS,ax
        mov     ax,(offs+ 6)[bp]
        mov     cs:WS,ax
        mov     ax,(offs+ 8)[bp]
        mov     cs:XD,ax
        mov     ax,(offs+10)[bp]
        mov     cs:YD,ax
        mov     ax,(offs+12)[bp]
        mov     cs:PG,al
        mov     ax,(offs+14)[bp]
        mov     cs:DR,al
else
        mov     ax,ds
        mov     es,ax
        mov     ax,(offs    )[bp]
        mov     cs:XS,ax
        mov     ax,(offs+ 2)[bp]
        mov     cs:YS,ax
        mov     ax,(offs+ 4)[bp]
        mov     cs:HS,ax
        mov     ax,(offs+ 6)[bp]
        mov     cs:WS,ax
        mov     ax,(offs+ 8)[bp]
        mov     cs:XD,ax
        mov     ax,(offs+10)[bp]
        mov     cs:YD,ax
        mov     ax,(offs+12)[bp]
        mov     cs:PG,al
        mov     ax,(offs+14)[bp]
        mov     cs:DR,al
endif
        call    parameter_chack
        cmp     cs:error_flag,1
        je      no_no
        call    _byteblt

no_no:call      mem_clear
        pop_reg         <ds,es,di,si,dx,cx,bx,ax>
        mov     sp,bp
        pop     bp
        ret             ; End of Main Routin

BYTEBLT         endp
;*************************************************************************
; ｂｙｔｅｔｂｌｔ関数のパラメータ
;*************************************************************************
;
;       byteblt ( xs,ys,hs,ws,xd,yd,pg,dr )
;
;       xs      =       Source X Position       (Word)
;       ys      =       Source Y Position       (Word)
;       ws      =       Data Width              (Word)
;       hs      =       Data Hight              (Word)
;       xd      =       Destination X Position  (Word)
```

```
;       yd          =       Destination Y Position  (Word)
;       pg          =       Page Serect             (Byte)
;       dr          =       Direction Flag          (Byte)
;
;*************************************************************************
;       DS = ソース セグメント
;       ES = デスティネーション セグメント
;       SI = ソース オフセット
;       DI = デスティネーション オフセット
;*************************************************************************
;       XS = X座標開始点ソース側（文字数計算　0－79）
;       YS = Y座標開始点ソース側（文字数計算　0－24）
;       WS = X方向の幅       （文字数計算　1－80）
;       HS = Y方向の幅       （文字数計算　1－25）
;       XD = X座標開始点デスティネーション側（文字数計算　0－79）
;       YD = Y座標開始点デスティネーション側（文字数計算　0－24）
;       PG = バッファ ページング ( 1 - 4 )
;       DR = ディレクション      ( 0 or 1 )
;                                          0 = 表示領域からバッファへ
;                                          1 = バッファから表示領域へ
;*************************************************************************
_byteblt        proc    near
        push_reg <ax,bx,cx,si,di,ds,es> ; 使用レジスタ退避
        ;-----------------------------------------------------------------
        cmp     byte ptr cs:DR,0
        je      crt_to_buffers
        cmp     byte ptr cs:DR,1
        je      buffers_to_crt
        if debug
        push    cs
        pop     ds
        mov     dx,offset dir_err_mes
        mov     ah,9
        int     21h
        endif
        call    mem_clear
        pop_reg <es,ds,di,si,cx,bx,ax>
        mov     ax,0001h        ; Direction Error = 0001
        ret
        ;-----------------------------------------------------------------
crt_to_buffers:                                 ; 表示領域からバッファーへ
        mov     ax,disp_seg
        mov     ds,ax                   ; DS = A000h
        mov     ax,cs
        mov     es,ax                   ; ES = CS
        xor     di,di                   ; DI = 0
        xor     si,si                   ; SI = 0

        cmp     cs:PG,1
        jne     aaat_2
        mov     di,offset PAGE_1_CRT
        jmp     job_2
aaat_2:         cmp     cs:PG,2
        jne     aaat_3
        mov     di,offset PAGE_2_CRT
        jmp     job_2
aaat_3:         cmp     cs:PG,3
        jne     aaat_4
        mov     di,offset PAGE_3_CRT
        jmp     job_2
aaat_4:         cmp     cs:PG,4
        jne     aaat_s
        mov     di,offset PAGE_4_CRT
```

```
        jmp     job_2
aaat_s:
        if debug
        push    cs
        pop     ds
        mov     dx,offset page_err_mes
        mov     ah,9
        int     21h
        endif
        call    mem_clear
        pop_reg <es,ds,di,si,cx,bx,ax>
        mov     ax,0002h        ; Direction Error = 0001
        ret
        ;---------------------------------------------------------------
buffers_to_crt:                                 ; バッファーから表示領域へ
        mov     ax,disp_seg
        mov     es,ax                   ; DS = A000h
        mov     ax,cs
        mov     ds,ax                   ; ES = CS
        xor     di,di                   ; DI = 0
        xor     si,si                   ; SI = 0

        cmp     cs:PG,1
        jne     next_2
        mov     si,offset PAGE_1_CRT
        jmp     job_2
next_2:         cmp     cs:PG,2
        jne     next_3
        mov     si,offset PAGE_2_CRT
        jmp     job_2
next_3:         cmp     cs:PG,3
        jne     next_4
        mov     si,offset PAGE_3_CRT
        jmp     job_2
next_4:         cmp     cs:PG,4
        jne     next_s
        mov     si,offset PAGE_4_CRT
        jmp     job_2
next_s:
        if debug
        push    cs
        pop     ds
        mov     dx,offset page_err_mes
        mov     ah,9
        int     21h
        endif
        call    mem_clear
        pop_reg <es,ds,di,si,cx,bx,ax>
        mov     ax,0002h        ; Direction Error = 0001
        ret
        ;---------------------------------------------------------------
job_2:                                          ; Set Addres Offset
        mov     ax,160                          ; SI = ( 160 * YS ) + XS
        mov     bx,cs:YS
        mul     bx
        add     ax,cs:XS
        add     ax,cs:XS
        add     si,ax
        ;---------------------------------------------------------------
        mov     ax,160                          ; DI = ( 160 * YD ) + XD
        mov     bx,cs:YD
        mul     bx
        add     ax,cs:XD
```

```
        add     ax,cs:XD
        add     di,ax
        ;-----------------------------------------------------------------
crt_out:
        mov     ax,ds:[si]                              ; Source Addres
        mov     es:[di],ax                              ; Destination Addres
        inc     si
        inc     si                                      ; Skip Word
        inc     di
        inc     di                                      ; Skip Word
        inc     cs:XCOUNT
        mov     bx,cs:WS
        cmp     cs:XCOUNT,bx
        jne     crt_out
        mov     ax,80                                   ;
        sub     ax,cs:WS
        mov     bx,2
        mul     bx
        add     si,ax
        add     di,ax
        mov     cs:XCOUNT,0
        inc     cs:YCOUNT
        mov     bx,cs:HS
        cmp     cs:YCOUNT,bx
        jne     crt_out

        cmp     cs:WORK_FLAG,1
        je      return_job

        mov     cs:WORK_FLAG,1
        cmp     cs:DR,1
        je      buf_to_att
        call    att_to_buffers
        mov     cs:XCOUNT,0
        mov     cs:YCOUNT,0
        mov     ax,attb_seg
        mov     ds,ax
        mov     ax,cs
        mov     es,ax

        jmp     job_2

buf_to_att:
        call    buffers_to_att
        mov     cs:XCOUNT,0
        mov     cs:YCOUNT,0
        mov     ax,attb_seg
        mov     es,ax
        mov     ax,cs
        mov     ds,ax
        jmp     job_2
        ;-----------------------------------------------------------------
return_job:
        call    mem_clear
        pop_reg <es,ds,di,si,cx,bx,ax>
        ret
_byteblt        endp
        ;-----------------------------------------------------------------
mem_clear       proc    near    ; Buffer Memory Clear
        mov     ax,cs
        mov     es,ax
```

```
        mov     di,offset XS
        mov     cx,20
        mov     ax,0
        cld
        rep     stosb
        gvram   0                       ; Select Gvram Page 0
        ret
mem_clear       endp

;*************************************************************************
; 属性アドレス取得サブルーチン           （属性領域からバッファーへ）
;*************************************************************************

att_to_buffersproc      near
        mov     ax,attb_seg
        mov     ds,ax                           ; DS = A200h
        mov     ax,cs
        mov     es,ax                           ; ES = CS
        xor     di,di                           ; DI = 0
        xor     si,si                           ; SI = 0

        cmp     cs:PG,1
        jne     bbbt_2
        mov     di,offset PAGE_1_ATT
        jmp     job_ret
bbbt_2:         cmp     cs:PG,2
        jne     bbbt_3
        mov     di,offset PAGE_2_ATT
        jmp     job_ret
bbbt_3:         cmp     cs:PG,3
        jne     bbbt_4
        mov     di,offset PAGE_3_ATT
        jmp     job_ret
bbbt_4:         mov     di,offset PAGE_4_ATT
job_ret:
        ret
att_to_buffersendp

;*************************************************************************
; 属性アドレス取得サブルーチン           （バッファーから属性領域へ）
;*************************************************************************

buffers_to_attproc      near
        mov     ax,attb_seg
        mov     es,ax
        mov     ax,cs
        mov     ds,ax
        xor     di,di                           ; DI = 0
        xor     si,si                           ; SI = 0

        cmp     cs:PG,1
        jne     ccct_2
        mov     si,offset PAGE_1_ATT
        jmp     att_ret
ccct_2:         cmp     cs:PG,2
        jne     ccct_3
        mov     si,offset PAGE_2_ATT
        jmp     att_ret
ccct_3:         cmp     cs:PG,3
        jne     ccct_4
        mov     si,offset PAGE_3_ATT
        jmp     att_ret
ccct_4:
```

```
        mov     si,offset PAGE_4_ATT
att_ret:
        ret
buffers_to_attendp

;***************************************************************************
; パラメータ・チェック
;***************************************************************************

parameter_chack         proc    near
        cmp     cs:xs,80
        jnb     error_1
        jmp     par_2
error_1:
        jmp     exit_1
par_2:
        cmp     cs:ys,25
        jnb     error_2
        jmp     par_3
error_2:
        jmp     exit_2
par_3:
        cmp     cs:hs,26
        jnb     error_3
        jmp     par_4
error_3:
        jmp     exit_3
par_4:
        cmp     cs:ws,81
        jnb     error_4
        jmp     par_5
error_4:
        jmp     exit_4
par_5:
        cmp     cs:xd,80
        jnb     error_5
        jmp     par_6
error_5:
        jmp     exit_5
par_6:
        cmp     cs:yd,25
        jnb     error_6
        jmp     par_7
error_6:
        jmp     exit_6
par_7:
        jmp     par_8
error_7:
        jmp     exit_7
par_8:
        cmp     cs:pg,4
        jnb     error_8
        jmp     par_9
error_8:
        jmp     exit_8
par_9:
        cmp     cs:dr,2
        jnb     error_9
        ret
error_9:
        jmp     exit_9

exit_1:         if      debug
```

```
        print   perror_1
        endif
        jmp     exit_0
exit_2:         if      debug
        print   perror_2
        endif
        jmp     exit_0
exit_3:         if      debug
        print   perror_3
        endif
        jmp     exit_0
exit_4:         if      debug
        print   perror_4
        endif
        jmp     exit_0
exit_5:         if      debug
        print   perror_5
        endif
        jmp     exit_0
exit_6:         if      debug
        print   perror_6
        endif
        jmp     exit_0
exit_7:         if      debug
        print   perror_7
        endif
        jmp     exit_0
exit_8:         if      debug
        print   perror_8
        endif
        jmp     exit_0
exit_9:         if      debug
        print   perror_9
        endif

exit_0:         if      debug
        print   keywait
        mov     ah,1
        int     21h
        endif
        mov     cs:ERROR_FLAG,1
        mov     dl,7
        mov     ah,2
        int     21h
        ret
parameter_chack         endp

;**********************************************************************
; バッファー エリア
;**********************************************************************

        XS              dW      ?       ; Source X Position
        YS              dW      ?       ; Source Y Position
        WS              dW      ?       ; Data Width
        HS              dW      ?       ; Data Hight
        XD              dw      ?       ; Destination X Position
        YD              dw      ?       ; Destination Y Position
        PG              db      ?       ; Page Serect
        DR              db      ?       ; Direction Flag
        XCOUNT          dw      ?       ; Direction X Counter
        YCOUNT          dw      ?       ; Direction Y Counter
        WORK_FLAG       db      ?
        ERROR_FLAG      db      ?
```

```
    PAGE_1_CRT      db      4000 dup(?)     ; Crt buffers Page 1
    PAGE_1_ATT      db      4000 dup(?)     ; Att buffers Page 1
    PAGE_2_CRT      db      4000 dup(?)     ; Crt buffers Page 2
    PAGE_2_ATT      db      4000 dup(?)     ; Att buffers Page 2
    PAGE_3_CRT      db      4000 dup(?)     ; Crt buffers Page 3
    PAGE_3_ATT      db      4000 dup(?)     ; Att buffers Page 3
    PAGE_4_CRT      db      4000 dup(?)     ; Crt buffers Page 4
    PAGE_4_ATT      db      4000 dup(?)     ; Att buffers Page 4

;****************************************************************
; メッセージ エリア
;****************************************************************

    if debug

perror_1    db      13,10,'ｘｓパラメータが範囲を越えてます．',13,10,'$'
perror_2    db      13,10,'ｙｓパラメータが範囲を越えてます．',13,10,'$'
perror_3    db      13,10,'ｈｓパラメータが範囲を越えてます．',13,10,'$'
perror_4    db      13,10,'ｗｓパラメータが範囲を越えてます．',13,10,'$'
perror_5    db      13,10,'ｘｄパラメータが範囲を越えてます．',13,10,'$'
perror_6    db      13,10,'ｙｄパラメータが範囲を越えてます．',13,10,'$'
perror_8    db      13,10,'ｐｇパラメータが範囲を越えてます．',13,10,'$'
perror_9    db      13,10,'ｄｒパラメータが範囲を越えてます．',13,10,'$'
keywait             db      13,10,'任意のキーを押して下さい．',13,10,'$'

    endif

    endps
    end
```

```
A>DIR/W
 ドライブ A: のディスクにはボリュームラベルがありません
 ディレクトリは A:¥

WINDOW   C     WINDOW  H     WCAT   C     BITBLT  ASM   BYTEBLT  ASM
CTS      LIB   LC      EXE   LC1    EXE   LC2     EXE   LC1B     EXE
LC2B     EXE   LCS     LIB   STDIO  H     LINK    EXE   CC       OBJ

       15 個のファイルがあります
   xxxxxx バイトが使用可能です

A>lc -ms -iA:¥ -iA:¥ wcat.c
Lattice MS-DOS C Compiler, Version 3.00
Copyright (C) 1985 Lattice, Inc.  All rights reserved.

Compiling wcat.c
        ワーニングエラーが出ますが実行には関係ありません
Module size P=0E79 D=004A U=027C

Total files: 1. Successful compilations: 1


A>link A:¥cc wcat,wcat,,A:¥lcs A:¥cts

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microcoft Corp 1983, 1984, 1985

Warning: no stack segment

A>
```

画面5-1-1 コンパイル手順

5-4で解説するexpert.cのコンパイル手順もついでに示しておきます.

```
A>DIR/W

 ドライブ A: のディスクにはボリュームラベルがありません
 ディレクトリは A:¥

LIST     C      EXPERT  C      CAR     RUL    LC      EXE    LC1    EXE
LC2      EXE    LC1B    EXE    LC2B    EXE    LCS     LIB    STDIO  H
LINK     EXE    CC      OBJ    CTYPE   H
        13 個のファイルがあります
   xxxxxx バイトが使用可能です

A>lc -ms -iA:¥ -iA:¥ expert.c
Lattice MS-DOS C Compiler, Version 3.00
Copyright (C) 1985 Latiice, Inc.  All rights reserved.

Compiling expert.c
         ワーニングエラーが出ますが実行には関係ありません
Module size P=05EB D=009A U=3F50

Total files: 1. Successful compilations: 1


A>link A:¥cc expert,expert,,A:¥lcs

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985

Warning: no stack segment

A>
```

5-1-2-6 プログラムの実行

サンプルプログラム〝WCAT.C〟はコンパイル後,〝WCAT.EXE〟になるので,

A：WCAT

で実行されます. 起動後はSelect windowで,

1： Show file　　2： Reshape

と聞いてくるので, 1か2を入力します.

1が入力された場合, マウス・カーソルが表示されます. プレス→領域設定→リリースで新しいウィンドウが開かれ, ファイルネームを聞いてきます. ソースファイルのファイル名を入力すると, 再度マウス・カーソルが表示され, 同様の手順でウィンドウをオープンすると, ファイルの内容がウィンドウ内に表示されます.

5-2 PC-9800シリーズ上でのAI研究プログラム

5-2-1 AI(Artificial Intelligence)とは？

人間の思考のプロセスを解明し，それを応用して，より知的なシステムを作ろうというのが人口知能(AI)研究の主な目的です．

初期のAIはチェスプログラムなどの研究が中心でしたが，80年代に入ってエキスパートシステムや機械翻訳システム等の実用システムも生まれてきています．

一方,心理的側面からの研究も盛んで,カーネギーメロン大学のアンダーソンのACT理論やエール大学のシャンクのCD理論などが展開されています．

最近，パソコンも16ビットが主流となり，LISP，Prolog，Smalltalk等，AI向きの言語処理系が動くようになってきています．ここではPC-9800シリーズ上でのAI研究の可能性を探ってみることにします．

AIは純粋AIと応用(Applied)AIとに分けることができ，その違いを以下に示します．

純粋AI：コンピュータを使って人間の思考の定式化を目指す学問

応用AI：AI研究の成果，副産物を産業的目的に応用する

エキスパートシステムや機械翻訳システムなどは応用AIの例です．応用AIは工学的ですが,純粋AIは逆に自然科学的，心理学的です．

5-2-2 AI向き処理系

AI構築用と言われている言語処理系としては，

・LISP

・Prolog

・Smalltalk

等があります．何もCやBASICでAI的プログラムが書けないという訳ではありませんが，やはりLISP，Prolog，SmalltalkがAI向きと呼ばれるには理由があります．それは柔軟な表現形式が可能であり，論理的，哲学的基礎にあるということです．

ここではLISPで「Waltz Lisp」，Prologで「PROLOG-KABA」，Smalltalkで「methods」の3つを紹介します．

5-2-2-1 Prolog

Prologは述語論理という理論形態から生まれた処理系で，フランスで開発されました．日本の第5世代計画(ICOT)での主言語となったことで一躍注目を浴びています．

Prologは後述のLISPと同様に方言がありますが，中心的なのは「DEC-10 PROLOG」の流れ

であるといえます．また，ICOTではPSI上で動くESPというProlog系の処理系を完成させています．

PROLOG-KABAを起動するには，>PROLOGと入力すると，PROLOG-KABAで書かれたemacs風のPrologエディタ(Pro-Edit)に入ります．

すでにプログラムしたファイルを読み込むにはCTRL-X, CTRL-Vとします．ファイル名をきいてくるので，入力するとそのファイルが読み込まれます．

セーブするときにはCTRL-X, CTRL-Wと入力し，ファイル名を入力してその名前のファイルに書き込みます．

読み込んだプログラムはreconsultするCTRL-X, CTRL-Sによってシステムに定義されます．このとき注意することはCTRL-X, CTRL-Sでは以前のユーザー定義述語はすべて消されているということです．

Pro-EditはPrologで書かれているため，カスタマイゼーションが非常に楽です．Prologが立ち上がるときには，INIT.PLというファイルを評価しますから，ここにユーザー述語を書いておけば，自分の慣れたエディタ風にすることができます．

reconsultした後にはPrologインタープリタによってプログラムを走らせます．Prologインタープリタへ行くにはCTRL-X, CTRL-Zと入力します．

!?-というプロンプトが出ているので，評価したい(走らせたい)述語を入力します．システムを終了させるときは!?-haltと入力します．

PROLOG-KABAはレベル1，2，3とあり，グラフィックスのサポートされているレベルではマルチウィンドウシステムも作れます(KABA-WINGではすでに実現されています)．

5-2-2-2 Smalltalk

LISP，Prologと共に注目されているのがオブジェクト指向言語であり，その代表格であるSmalltalk-80の処理系です．

オブジェクト指向という考え方はHewittという人のアクターモデルに基づいていますが，処理系としては1972年頃からXEROXのPARC(パロアルトリサーチセンター)で試作され発展したSmalltalk系が有名です．

オブジェクト指向というのは今までの，処理手続きの記述を中心にしたのとは別に，存在物つまりデータを中心とした計算モデルです．

例えば，1というオブジェクト(存在物)は整数というクラス(グループ)であり，整数の演算規則に従うというような考え方です．ここでは1をインスタンスといいます．

演算はメッセージパッシングという方法で行います．

1に対して〝+1〟というメッセージは〝+→加算〟であり，その引数は1となります．1というインスタンスはこのメッセージを受け取ると1自身では加算法を知らないので，上位クラスの整数へ行きます．整数自身も特別な加算手続き(メソッド)を定義していないときは，さらにその

上の数というクラスに行き，＋の方法を得ます．この結果1+1が行われ，2というインスタンスが生成されて処理を終ります．

このように書くとややこしいようですが，この考え方を理解すると非常にすっきりとプログラムが作れます．

多くのウィンドウシステムは各ウィンドウを1つのオブジェクトとして作られているように，複雑なシステムを作る際に特に強力です．

methods はマルチウィンドウを使っていますが，これはバイト単位のウィンドウシステムで5-1-2で述べたBYTEBLT を中心にしています．

マウスを動かすとカーソルが動きます．work space というのがいわゆるエディタに相当する部分で，ここにプログラムを書きます．

browser はシステムのオブジェクトを探し，内容を見るためにあります．work space に入力したプログラムを実行させるには，マウスで左ボタンを押しながらセレクトしてリバース状態にし，右ボタンをクリックします．メニューがでたら do it を選びます．ただし，結果がディスプレイされるようにプログラムされていないと何も分からないので，何か変化を見たい人は show it がてっとり早いでしょう．

また，ウィンドウの端で左ボタンをクリックすると，ウィンドウコントロールメニューが出ます．必要なら1つ選んでウィンドウの状態を変えます．

システムから出るときは，バックグラウンドで左ボタンをクリックするとシステムメニューが出ます．

今の状態をセーブしたい時は save image してから，セーブしない時はそのまま exit methods を選びます．また，このメニューではファイルディレクトリのブラウジングができる brows disk があります．

open work space を選ぶと新しい作業場所が作られます．メニューの選択は左ボタンのクリックです．

brows disk を選ぶとディスク名の入力を要求してくるので，A などと入力します．そして左クリックすると accept とか cansel とか書いたメニューが出るので A でよいなら accept を選び，否なら cansel します．

exit methods を選ぶと forget image ，continue ，saving image を選ぶメニューが出るので1つ選びます．continue 以外はシステムへ戻ります．

5-2-2-3　LISP

LISP は1960年代に MIT の McCathy によって提唱された Λ(ラムダ)計算に基づいた処理系です．アメリカにおける AI は主に LISP を用いてなされています．

LISP には初期 LISP 1.5 から始まって MAC LISP 系と INTER LISP 系がありますが，1つの処理系の中での方言の問題を解決するために最近 common lisp が標準 LISP として提唱され，

PC-9800 シリーズ上でも動くシステムがあります.

しかし，common lisp にも少々，方言が出回り始めています．次に LISP を使った AI 研究用プログラムを紹介します.

ここでは Procode International 社の Waltz Lisp を取り上げます．Waltz Lisp は UNIX 上で標準的な Frantz Lisp とほとんど同じ仕様を持った Lisp の処理系です.

起動法は A：WLS で行いますが，プログラムファイルの読み込みを立ち上げと同時に行うには，

```
A>WLS ファイル名
```

とします．さらに，実行する関数名も入力できます．例えば，

```
A>WLS CA.L (CA '(JOHN GAVE MARY A BOOK))
```

と入力すれば，Lisp が起動と同時に〝CA.L〟が読み込まれ，さらに(CA '(JOHN GAVE MARY A BOOK))が評価(実行)されます.

Waltz Lisp はデバッグ機能もあり，関数の呼び出しをモニタする trace や，ある関数へ行くときに実行を停止する tbreak などがあります．例えば，プログラムをロードした後，

```
>(trace match findp)              → match と findp をトレース
t                                 ← (trac…)の評価が成功しトレース状態にした
>(tbreak watch findp follow)      → match findp follow でブレーク
t
```

とすると，関数 match と findp を呼び出すときに，その関数名と引数の内容が表示され，関数から脱出するときに，その値が示されます.

ブレークポイントは match findp follow において発生します. break からの脱出は t で実行が再開されます.

Lisp のプログラミング方法に関しては，すでに多くの書物が出ているのでここでは省略しますが，Waltz Lisp を使う方には Robert Wilensky によって書かれた LispCraft という本を推薦します．ただし，翻訳はされていません.

Common lisp を使うには Golden Common Lisp が PC-9801 上で動きますが，これを十分に動作させるには 640 KB のメモリの実装が必要でしょう.

5-3　自然言語処理

5-3-1　ELIZA

自然言語処理(NLP)とは，生の英文，和文をコンピュータに理解させる試みです．現在も多くの研究が続けられています．本項では自然言語処理の古典的研究であるELIZAプログラムのC言語版を紹介します．

ELIZAは1966年に，MIT(マサチューセッツ工科大学)のJ. Weizenbaum教授によって発表された，初期の自然言語処理システムです．

ELIZAは極めて単純かつ表面的な処理だけであたかもプログラムが会話をしているように見えるシステムです．

このシステムは，入力文の中のキーワードを発見して，それに対応したテンプレートに入力文を当てはめて応答文を生成する，テンプレート・マッチングと呼ばれる手法を用いています．

ELIZAは，いかなる意味でも入力文を理解している訳ではありませんが，時としてその実力以上にリアルな会話を行うことがあります．

このプログラムはDOCTORとも呼ばれており，多くのLISP処理系でサンプルとして取り上げられています．

オリジナルのELIZAはMAD-Slipという LISPの一種で記述されていますが，ここではC言語で記述します．

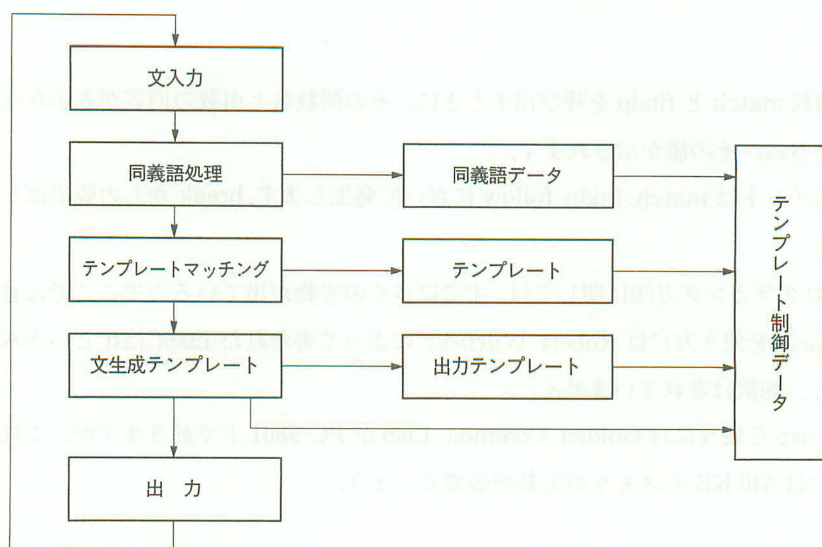


図5-3-1　システム構成

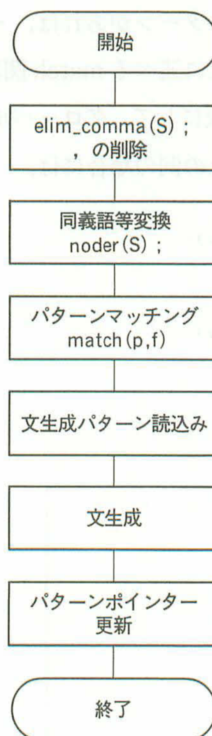


図 5-3-2　ELIZA フローチャート

5-3-1-1　同義語処理

これは例えば，NEC，IBM，APPLE などというように，似た意味の単語をまとめることです．この例では Computer-maker とする等の処理を行います．この処理を行うことによって，入力文の受け入れられる範囲が広くなります．

データは，リストまたは文字列の形になっていて，入力文の中の単語にマッチするものがあると，その対の変換単語に置き換えます．

入力文　：　I know LISP

に対して，PROGRAMING-LANG LISP PROLOG SMALLTALK LOOPS という対がマッチするので，変換語〝PROGRAMING-LANG〟に置き変わり，

I know programing-lang

となります．

5-3-1-2　パターンマッチャー

入力文に対して用意されたパターンを逐次トライして，マッチするパターンを探してそのスロットを埋めます．

入力文　：　I love you

に対して，〃*0 I *1 you *0〃というパターンがあれば，*1にはloveが入って入力はこのパターンにマッチします．これらの処理は次に述べるmatch関数を使って行います．

match関数はパターンと入力文を引数にして，グローバル変数で定義されているstackの中にマッチした文字列を入れる関数です．上の例の場合には，

stack[0] → 〃〃 ……何も入っていない
stack[1] → 〃Love〃
stack[2] → 〃〃 ……何も入っていない

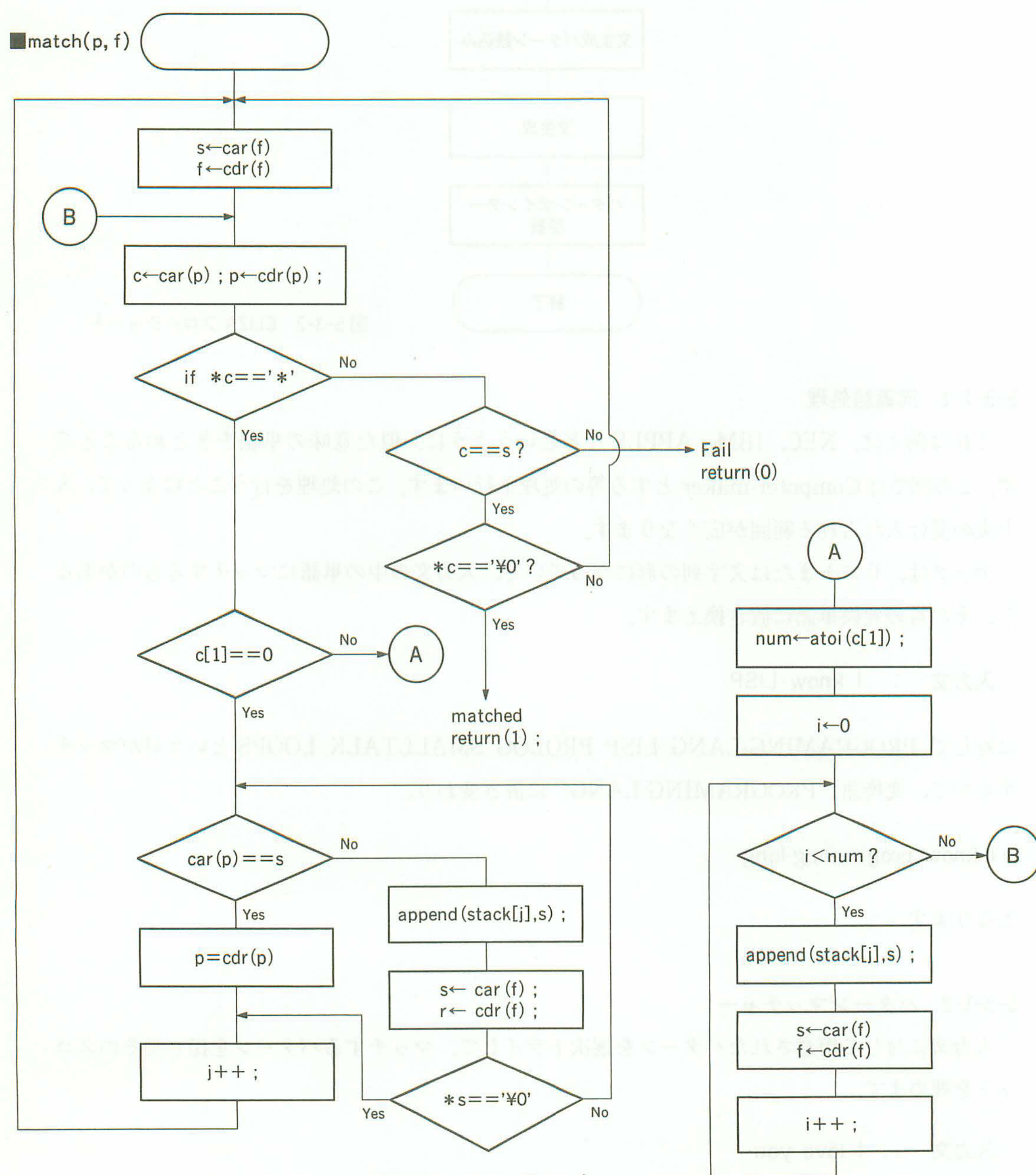


図5-3-3　フローチャート

となります．＊0は無制限のマッチで，＊1は1単語のマッチです．ゆえに，上のパターンに対して〝I really love you〟はマッチしません．この場合には，＊0 I ＊2 you ＊0というパターンが必要です．

5-3-1-3　出力生成部

パターンマッチャーで埋められたスロットとマッチしたパターンのキーを基に出力生成テンプレートを決定して，出力文を作ります．

replydataにはキーと生成テンプレート数が記述されています．

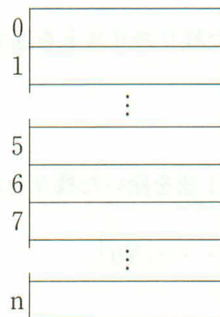


図 5-3-4　出力生成テンプレート

たとえば

キー：05
生成テンプレート数：03

の場合は，キー05のテンプレートから07までが順次生成に使われます．

5-3-1-4　文字列操作関数

ここでは，C言語上にLISPの組み込み関数のいくつかを疑似的に実現します．

C言語を使って自然言語，AIのシステムを作成する際に，避けて通れない文字列操作を容易にするための関数です．

Cコンパイラの中には，強力な文字列操作が提供されているものもありますが，ここではすべてのCコンパイラで使用可能にするために，コンパイラ独自の拡張関数は使用していません．

関数名はLISPに準拠しています．CでLISP的書法を用いる場合や，LISPに慣れた人には分かり易いと思われます．

1. car

carはLISPで，リストの先頭の要素を取り出す関数です．

(car '(A B C))はAとなります．

C言語の関数として文字列操作に応用するなら，リストを文字列とし，その第1語を取り出す関数と定義します．つまり，文字列をサーチし，最初の空白までを返す関数とします．

A	B	C		D	E	・・・・・・・		¥0

に car を作用させると，

A	B	C	¥0

が返されます．

2．cdr

cdr は car とは逆にリストの先頭を除いた残りのリストを返す関数です．

(cdr '(A B C))は(B C)となります．

C でのインプリメントでは，文字列の第1語を除いた残りを返す関数と定義します．

A	B	C		D	E		F	・・・・・・・	¥0

に cdr を作用させると，

D	E	F		・・・・・・・	¥0

が返されます．

3．append

append は，2つのリストを結合する関数です．

(append '(A B) '(C D))は(A B C D)となります．

C では2つの文字列の結合とします．ただし，文字列の中の要素の間には空白を1つ入れているので，strcat(a,b)では strcat("abc"，"def")；→ "abcdef" となってしまいます．append ("abc"，"def")；→ "abc def" とするために，append されるリストに文字列がすでに存在するかどうかをチェックし，存在するときには空白を加えた後に，strcat する方法を取っています．

4．del

del は LISP の組み込み関数ではありませんが，文字列操作において特に有用なので定義することにします．

del はリスト中の要素の中で，特定のものを削除する関数です．

(del 'a '(abc))は(bc)を返します．

Cでは文字列の中の単語が指定され語と一致したら，その単語を削除した文字列を返します．

"A BC DE F"

A		B	C		D	E		F	¥0

削除 "BC"

B	C	¥0

del を作用させると，

"A DE F"

A		D	E		F	¥0

を返します．

5. member

member は対象となるリストの中に指定された要素が存在すると真(1)を返し，存在しなければ，偽(0)を返す関数です．

```
(member 'a '(a b c))  → t.
(member 'a '(b c d))  → nil.
```

C言語では文字列中に指定された単語が発見されたら1を返し，ないときは0を返す関数とします．

```
member("a", "a b c")→ 1
member("a", "b c d")→ 0
```

6. kread

kread は文字列を読み込む関数です．この関数ではコンソールから文字列を読み込み，以下の変換を行います．

・すべて小文字に変換
・空白の連続を1つの空白に変換
・？！等を．に変換

プログラム5-3-1　ELIZA (eliza.c)

```
/* eliza.c                                                          */
/*                              ELIZA                  version 3.11 */
/*                  update 1985.02.08 copyright (C) DMSC            */
/*          (Lattice C version of ELIZA defined by Dr.Wizenbaum)    */

#include "stdio.h"
#include "ctype.h"

char stack[4][80];

main(){
char s[80];
      for(;;) {
              strcpy(s,kread());
              eliza(s);
              }
}

eliza(s)
char s[80];
{
char outbuf[160];
static int replydata[44][2] = {

      { 0, 0 },         /* strline location, maximum number of replies */
      { 1, 4 },         /* do i remenber */
      { 5, 6 },
      {11, 4 },
      {15, 3 },         /* xxx */
      { 18, 4 },
      { 22, 1 },
      { 23, 5 },
      { 28, 4 },
      { 32, 2 },
      { 34, 6 },        /* 10 */
      { 40, 4 },
      { 44, 4 },
      { 48, 4 },
      { 52, 4 },
      { 56, 5 },        /* my */
      { 61, 2 },        /* was you */
      { 63, 5 },
      { 68, 6 },        /* was i */
      { 74, 8 },        /* you are sorry */
      { 82, 4 },
      { 86, 3 },
      { 89, 4 },        /* you belive */
      { 93, 4 },        /* you are */
      { 97, 4 },        /* you can't */
      {101, 4 },        /* you feel */
      {105, 4 },        /* you 0 i */
      {109, 7 },        /* i are  */
      {116, 3 },        /* i 0 you */
      {119, 4 },        /* i */
      {123, 4 },        /* yes */
      {127, 4 },        /* no */
      {131, 4 },        /* your family */
      {135, 9 },        /* your */
      {144, 4 },        /* can i */
      {148, 6 },        /* what */
      {154, 7 },        /* why can't you */
      {161, 4 },
```

```
        {165, 8 },
        {173, 4 },
        {177, 3 },
        {180, 4 },
        {184, 4 },
        {188, 1},
};

static int variate[44];       /* current variation for each type of reply */
int keyid;                   /* id number for each keywords */
int lineid;                   /* id number for reply lines */
char strline[160];
char keyword[44][25];
strcpy(keyword[0],"9");
strcpy(keyword[1],"*0 do I remember *0");
strcpy(keyword[2],"*0 you remember *0");
strcpy(keyword[3],"*0 always *0");
strcpy(keyword[4],"*0 you dreamt *0");
strcpy(keyword[5],"*0 dream *0");
strcpy(keyword[6],"*0 hello *0");
strcpy(keyword[7],"*0 perhaps *0");
strcpy(keyword[8],"*0 sorry *0");
strcpy(keyword[9],"*0 name *0");
strcpy(keyword[10],"*0 computer *0");
strcpy(keyword[11],"*0 are you *0");
strcpy(keyword[12],"*0 are I *0");
strcpy(keyword[13],"*0 why don't I *0");
strcpy(keyword[14],"*0 my *0");
strcpy(keyword[15],"*0 was you *0");
strcpy(keyword[16],"*0 you was *0");
strcpy(keyword[17],"*0 was I *0");
strcpy(keyword[18],"*0 you want *0");
strcpy(keyword[19],"*0 you are sad *0");
strcpy(keyword[20],"*0 you are happy *0");
strcpy(keyword[21],"*0 you believe you *0");
strcpy(keyword[22],"*0 you are *0");
strcpy(keyword[23],"*0 you can't *0");
strcpy(keyword[24],"*0 you feel *0");
strcpy(keyword[25],"*0 you *0 I *0");
strcpy(keyword[26],"*0 I are *0");
strcpy(keyword[27],"*0 I *0 you");
strcpy(keyword[28],"*0 can I *0");
strcpy(keyword[29],"*0 I *0");
strcpy(keyword[30],"*0 yes *0");
strcpy(keyword[31],"*0 no *0");
strcpy(keyword[32],"*0 your *0 family *0");
strcpy(keyword[33],"*0 your *0");
strcpy(keyword[34],"*0 what *0");
strcpy(keyword[35],"*0 why can't you *0");
strcpy(keyword[36],"*0 everyone *0");
strcpy(keyword[37],"*0 if *0");
strcpy(keyword[38],"*0 equal *0");
strcpy(keyword[39],"*0 you don't *0");
strcpy(keyword[40],"*0 can you *0");
strcpy(keyword[41],"*0 because *0");
strcpy(keyword[42],"*0 are *0");
strcpy(keyword[43],"*0");
/* eliza.c body */
      strcpy(s,elim_comma(s));
      if (strlen(car(s)) >= 20) {
              printf("OOPS!!! ¥n");
              return(0);
              }
```

```
        strcpy(s,noder(s));
    for (keyid=1; ((match(keyword[keyid],s) != 1) && (keyid < 44)); keyid++)
                                ;

        /* generate answer */
                lineid = replydata[keyid][0] + variate[keyid];
                strcpy(strline,getdis("eliza.txt",lineid));
                if (keyid == 43)
                        sprintf(outbuf,strline,stack[0]);
                else
                        sprintf(outbuf,strline,stack[1],stack[2]);

                variate[keyid]++;
                if (variate[keyid] == replydata[keyid][1])
                        variate[keyid] = 0;
                strcpy(outbuf,topup(outbuf));
                printf("%s ¥n",outbuf);

}


toucase(s)
/* chage a string into uppercase */
char s[];
{
int i;
      for ( i=0;(s[i] = toupper(s[i])) != '¥0'; i++ ) ;
      return (s);
}


elim_comma(s)
char *s;
{
int i,j;
      i = j = 0;
      while (s[i] != '¥0') {
              if (s[i] == ',') s[i++];
              s[j++] = s[i++];
      }
      return(s);
}


/* lindis.c */

getdis(fname,p)
/* return p th line of fname as a string */
char fname[];
int p;
{
int i,j;
long l;
      FILE *fp, *fopen();

      char linbuf[80];
long k;
      if ((fp=fopen(fname,"r"))==NULL) {
              printf("Pattern template file is not installed.¥n");
              return(0);
              }
```

```
k = 64;
l = k * p;
      fseek(fp,l,0);
      fgets(linbuf,80,fp);
        fclose(fp);
      i = j = 0;
      while (linbuf[i] != '¥n') {
              linbuf[j++] = linbuf[i++];
              }
      linbuf[j]='¥0';
      return(linbuf);
}



comember(s,t)
/* check whether two strings have a same item in common */
/* if yes: return the word.  if no: return null */
char s[],t[];
{
char u[80], v[80];
      strcpy(u,car(t));
      strcpy(v,cdr(t));
      if (member(u,s)) return(u);

      else
              if (strlen(v)==0) return(v);
              else comember(s,v);
}



/* noder.c */
/* changes the words of same nodes to a representative word */

noder(s)
char s[];
{
int maxnode;
char t[80],v[80];
static char *nodbuf[30] = {
"family mom dad brother sister mother father wife",
"are am",
"you i me",
"I you",
"computer computers crt dec ibm nec apple",
"I'm you're",
"languages lisp prolog pascal fortran udl",
"were was",
"was were",
"friends colleague friend companion pal",
"myself yourself",
"yourself myself",
"your my",
"my your",
"sad unhappy depressed",
"happy grad elated better",
"can't cannot",
"what how when",
"believe feel think belief wish",
"equal alike same",
"everyone noone nobody everybody",
"perhaps maybe",
```

```
"spell curse voo-doo hypnotize",
"charm becharm enchant fascinate glamour",
"dream fantasy illusion imagine fancy",
"request ask beg demand appeal plea",
"joke jest jape fun pun ridicule wit",
"reach come arrive land accomplish",
"expose show reveal uncover strip unveil",
"yes certainly sure right true aye ok o.k.",
};

int i, j;
maxnode = 30;
strcpy(t,"");
strcpy(v,"");
      strcpy(t,car(s)); strcpy(s,cdr(s));
      while (*t != '¥0') {
              j = 0;
              for (i = 0; j == 0 && i < maxnode; i++) {
                      if (member(t,cdr(nodbuf[i])) == 1){
                              strcpy(t,car(nodbuf[i]));
                              j = 1;
                              }
                      }
                      strcpy(v,append(v,t));
                      strcpy(t,car(s)); strcpy(s,cdr(s));
              }
      return(v);
}



topup(s)
char *s;
{
      s[0] = toupper(s[0]);
      return(s);
}

#include "list.c"

match(p,g)
char p[80], g[80];
{
int i, j, num, n;
char s[80], c[80], f[80], nbuf[3];

strcpy(f,g);
j = 0;
      for(n = 0; n < 4;n++) strcpy(stack[n],"");

loop1:
      strcpy(s,car(f));
      strcpy(f,cdr(f));
loop2:
      strcpy(c,car(p));
      strcpy(p,cdr(p));

      if( *c == '*' ) {
              if ( c[1] == '0' ) {
loop3:
                      if(strcmp(car(p),s) != 0) {
                              strcpy(stack[j],append(stack[j],s));
                              strcpy(s,car(f));
```

```
                                strcpy(f,cdr(f));
                                if ( *s != '¥0' ) {
                                        goto loop3;
                                        }
                                else {
                                        goto loop4;
                                        }
                                }
                        strcpy(p,cdr(p));
loop4:
                        j++;
                        goto loop1;
                        }
                else {
                        nbuf[0] = c[1]; nbuf[1] = '¥0';
                        num = atoi(nbuf);
                        for ( i = 0; i < num ; i++) {
                                strcpy(stack[j],append(stack[j],s));
                                strcpy(s,car(f));
                                strcpy(f,cdr(f));
                                }
                        goto loop2;
                        }
                }
        else {
                if ( strcmp(c,s) == 0 ) {

                        if ( *c == '¥0' ) {
                                return(1);
                                }
                        else {
                                goto loop1;
                                }
                        }
                else {
                        return(0);
                        }
                }
}
```

(list.c)

```
/*                                                              */
/*              Lattice C --- lisp-like functions ---           */
/*                update 1984 copy right (C) DMSC               */
/*                                                              */
/*      This function package include following lisp functions: */
/*              kread                                           */
/*              car                                             */
/*              cdr                                             */
/*              member                                          */
/*              mapcar                                          */
/*              del                                             */
/*              append                                          */
/*              cons                                            */

kread()
{
int pshift;
char readbuffer[80], c;
pshift = 0;
```

```
        printf("¥n>");
        while ((c = getchar()) != '¥n' ) {
                c = isupper(c) ? tolower(c) : c ;
                *(readbuffer+pshift) = c;
                switch (c) {
                        case '.':
                        case ',':
                        case '!':
                        case '?':
                                *(readbuffer+pshift) = ' ';
                                pshift++;
                                *(readbuffer+pshift) = ',';
                                break;
                        case ' ':
                                if (*(readbuffer+pshift-1) == ' ')
                                        pshift--;
                                break;
                        default:
                                break;
                        }
                         if (pshift < 77) pshift++;
                        else pshift = 78;
                }
                readbuffer[78] = '¥0';
                readbuffer[pshift] = '¥0';
                if(readbuffer[pshift-1] !=',')
                readbuffer[79] = '¥0';
                return(readbuffer);
}


car(s)
char *s;
{
char carbuffer[80],*t;
int pshift;
t = &carbuffer;
        pshift = 0;
        while(*(s+pshift) != ' ' && *(s+pshift) != '¥0' && pshift < 78) {
                *(t+pshift) = *(s+pshift);
                pshift++;
                }
                *(t+pshift) = '¥0';
        return(t);
}

cdr(s)
char *s;
{
char cdrbuffer[80], *t;
int pshift, tpshift;
t = &cdrbuffer;
        pshift = 0;
        tpshift = 0;
        while(*(s+pshift) != ' ' && *(s+pshift) != '¥0')
                pshift++;
        if (*(s+pshift) == ' ') {
        pshift++;
        while(*(s+pshift) != '¥0') {
                *(t+tpshift) = *(s+pshift);
                tpshift++;
                pshift++;
                }
```

```
        }
        *(t+tpshift) = '¥0';
        *(t+79) = '¥0';
        return(t);
}


member(s,t)
char *s,*t;
{
        int true,pshift;
        true=0;
        pshift = 0;
        while(*t != '¥0' && true == 0) {
                while(*(s+pshift) == *t && *t != '¥0') {
                        pshift++;
                        t++;
                }
                if (*t != ' ' && *t != '¥0') {
                        while (*t != ' ' && *t != '¥0')
                                t++;
                }
                else if (*(s+pshift) == '¥0') {
                        true = 1;
                }
                if (*t != '¥0') t++;
                pshift=0;
        }
        return(true);
}

mapcar(u,s,t)
char *u,*s,*t;
{
int pshift,tpshift,upshift;
tpshift =0, upshift=0;
        while(*(t+tpshift-1) != '¥0') {
                pshift = 0;
                while(*(s+pshift) == *(t+tpshift) && *(t+tpshift) != '¥0') {
                        pshift++;
                        tpshift++;
                        }
        if((*(t+tpshift)==' '||*(t+tpshift)=='¥0')&&*(s+pshift)=='¥0') {
                        *(u+upshift) = 'T';
                        }
                else {
                        *(u+upshift) = 'N';
                        while(*(t+tpshift) != ' ' && *(t+tpshift) != '¥0')
                                tpshift++;
                        }
                upshift++;
                tpshift++;
        }
        *(u+upshift) = '¥0';
}

append(s,t)
char *s,*t;
{
        if (strcmp(s,"")!=0)
        strcat(s," ");
        strcat(s,t);
        return(s);
}
```

```
cons(s,t)
char *s, *t;
{
char u[80];
int i , j;
i = j = 0;
        while((u[i] = *(s+i))!= '¥0' && i < 79)
                i++;
        if (u[0] != '¥0' && i < 79)
        u[i++] = ' ';
        while ((u[i] = *(t+j))!= '¥0' && i < 79) {
                i++;
                j++;
        }
        u[i] = '¥0';
        strcpy(t,u);
        return(t);
}

del(s,t)
char *s,*t;
{
char u[80];
int true;
strcpy(u,"");
        while (strcmp(car(s),"¥0") !=0) {
                true=strcmp(car(s),t);
                if (true !=0)
                strcpy(u,append(u,car(s)));
                strcpy(s,cdr(s));
        }
        strcpy(s,u);
        return(s);
}
```

(eliza.txt)

```
T
You mentioned %s. %0s
What about %s? %0s
Why do you think I should recall %s now?
Did you think I would forget %s? %0s
What is the connection between me and %s? %0s
What in the present situaton reminds you of %s? %0s
why do you remenber %s just now?
what else do you remenber?
does thinking of %s bring anything else to mind?
do you often think of %s?
really %s?
what do you think about %s?
do you wish that %s?
do you think its likely that %s?
have you dreamt %s before?
have you ever fantasied %s while you are awake?
really, %s?
don't you believe that dream has something to do with you?
what person appear in your dreams?
do you dream often?
what does that dream suggest to you?
how do you do.
you don't know.
```

```
you aren't sure.
can't you be more positive?
why the certain tone?
you don't seem quite certain.
please don't apologize.
i've told you that apologies are not necessary.
what feeling do you have when you apologize?
apologies are not necessary.
i've told you before that I am not interested in names.
i am not interested in names.
what do you think about artficial intelligence.
does artificial intellingence worry you?
don't you think computers can help Terrans?
what do you think of the machines that can think?
why do you mention computers?
do computers worry you?
do you believe you are %s?
would you want to be %s?
you wish I would tell you you are %s?
what would it mean if you were %s?
why are you interested in whether I am %s or not?
would you prefer if I weren't %s?
perhaps I am %s in your fantazies.
do you sometimes think I am %s?
did you think they might not be %s?
would you like it if they were not %s?
what if they were not %s?
possibly they are %s?
why are you concerned over my %s?
what about your own %s?
are you worried about someone else's %s?
really, my %s?
what if you were %s?
do you think you were %s?
were you %s?
what would it mean if you were %s?
what does ' %s ' suggest to you?
why do you tell me you were %s?
perhaps I already knew you were %s.
would you like to believe I was %s?
what suggests that I was %s?
what do you think?
perhaps I was %s.
what if I had been %s?
what would it mean to you if you got %s?
why do you want %s?
suppose you got %s soon.
what if you never got %s?
what would getting %s mean to you?
what does waiting %s have to do with this game?
i am sorry to hear you are %s.
do you think coming here will help you not to be %s?
i am sure it is not pleasant to be %s.
can you explain what made you %s?
how have I help you to be %s?
has your treatment made you to %s?
what makes you %s just now?
can you explain why you are suddenly %s?
how have I helped to be %s?
has your treatment made you %s?
what makes you %s just now?
can you explain why you are suddenly %s?
do you really think so?
but you are not sure you %s.
```

```
do you really doubt you %s?
is it because you are %s that you play this game?
how long have you been %s?
do you belive it normal to be %s?
do you enjoy being %s?
how do you know you can't %s?
have you tried?
perhaps you could %s now.
do you really want to be able to %s?
tell me more about such feeling.
do you often feel %s?
do you enjoy feeling %s?
of waht does feeling %s reminds you?
perhaps in your fantasy we %s each other.
do you wish to %s me?
you seem to need to %s me.
do you %s anyone else?
what makes you think I am %s?
does it please you to believe I am %s?
do you sometimes wish you were %s?
perhaps you would like to be able to %s.
why do you think I %s you?
you like to think I %s you.
what makes you think I %s you?
really, I %s you.
do you wish to believe I %s you?
suppose I did %s you, what would that mean?
does someone else believe I %s you?
you belive I can %s don't you?
you want to be able to %s.
perhaps you would like to be able to %s yourself.
we were discussing you, not me.
oh, I %s!
you are not talking about me aren't you?
what are your feeling now?
you seem quite positive.
you are sure.
i see.
i understand.
are you saying 'no' just to be negative?
you are being a bit negative.
why not?
why 'no'?
tell me more about your family.
who else in your family %s?
your %s.
what else comes to your mind when you think of your %s?
your %s?
why do you say your %s?
do that suggest any thing else which belong to you?
is it important to you that %s?
why do you ask?
does that question interest you?
what is it you really want to know?
are such questions much on your mind?
what answere would please you most?
what do you think?
what comes to your mind when you ask that?
have you ask such question before?
have you asked anyone before?
do you think you should be able to %s?
do you want to be able to %s?
do you think this would help you to %s?
have you any idea why you can't %s?
```

```
really %s?
surely not %s.
can you think of any one paticular?
who, for example?
you are thinking of very special person.
who, may I ask?
someone special, perhaps.
you have particular person in mind don't you?
who do you think you are talking about?
when you think of a specific example?
when?
what incident are you thinking of?
really, always.
in what way?
what resemblance do you see?
what does that similarity suggest you?
what other connections do you see?
what do you suppose trhat resemblance means?
what is the connection, do you support?
could there really be some connection?
how?
won't you really %s?
why don't you %s?
do you wish to be able to %s?
does that trouble you?
whether or not you can %s depends on you more than on me.
do you want to be able to %s?
perhaps you don't want to %s.
is that the real reason?
don't any other reasons come to mind?
does that reason seem to explain anything else?
what other reasons might there be?
do you believe I don't %s?
perhaps I will %s in good time.
should you %s yourself.
you want me to %s.
you mention %s.

/*
        一行が64バイトになるようにリフォームする必要があります。
        このテキストファイルの全行を64バイトにするためには、付録の
        FORM64.Cをお使い下さい。
*/
```

```
A>DIR/W

 ドライブ A: のディスクにはボリュームラベルがありません
 ディレクトリは A:¥

ELIZA    C      LIST     C      ELIZA    TXT    LC       EXE    LC1      EXE
LC2      EXE    LC1B     EXE    LC2B     EXE    LCS      LIB    STDIO    H
LINK     EXE    CC       OBJ    CTYPE    H      FORM64   C

        14 個のファイルがあります
    xxxxxx バイトが使用可能です

A>lc -ms -iA:¥ -iA:¥ form64
Lattice MS-DOS C Compiler, Version 3.00
Copyright (C) 1985 Lattice, Inc.  All rights reserved.
```

```
Compiling form64.c
        ワーニングエラーが出ますが実行には関係ありません
Module size P=00FF D=002E U=0000

Total files: 1. Successful compilations: 1


A>link A:¥cc form64,form64,,A:¥lcs

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985

Warning: no stack segment

A>REN ELIZA.TXT ELIZA.DOC

A>FORM64 ELIZA.DOC ELIZA.TXT

A>lc -ms -iA:¥ -iA:¥ eliza
Lattice MS-DOS C Compiler, Version 3.00
Copyright (C) 1985 Latiice, Inc.  All right reserved.

Compiling eliza.c
        ワーニングエラーが出ますが実行には関係ありません
Module size P=0C90 D=0669 U=0198

Total files: 1. Successful compilations: 1

A>link A:¥cc eliza,eliza,,A:¥lcs

Microsoft 8086 Object Linker
Version 3.00 (C) Copyright Microsoft Corp 1983, 1984, 1985

Warning: no stack segment

A>
```

画面5-3-1 コンパイル手順

5-3-1-5 実行方法

ELIZAを実行するには，〝ELIZA.EXE〟と〝ELIZA.TXT〟がAドライブのディスクにあることを確認して，A>ELIZAと入力することによって起動します．

プロンプト〝>〟が出ているときに，英文を入力しリターンすると応答文が返ってきます．実行例を次図に示します．

```
A>ELIZA

>Good morning!
 I will good morning in good time.

>What do you think about me?
 you were believe about?

>Do you like me?
Ke to be able to like.

>I love you!
 of waht does feeling love reminds you?
```

```
>.....
 I will ..... in good time.

>bye bye
 I will bye bye in good time.
>ｱﾉﾈｰ
 I will ｱﾉﾈｰ in good time.

>^C
```

画面5-3-2 ELIZA実行例

5-3-2 AI on PC with LISP

AIシステムの構築にはLISP，Prologが多く使われます．ここではLISPによるAI研究プログラムを紹介します．

システムは自然言語理解の研究のためのプログラムでConceptual Analyser(概念解析プログラム)と呼ばれ，Inside Computer Understandの第13章のLawrence BirnbaumとMallory Selfridgeの論文の中でその作り方が示されているものです．

このプログラムは英語を入力し，それをCDformという，言語に依存しない知識表現の形式に変換するものです．

例えば，MOVEやFLYはすべて物理的移動であるPTRANSという構造になります．MOVEとFLYの区別は何がその主体(ACTOR)かという言語外の知識によって決定される部分とPTRANSの構造の中で示される部分とに別れます．

この考え方はYale大学のRoger Schank教授によって提唱され，AI研究に大きな影響を与えています．

AI研究の1つのアプローチに人間の認知過程のシュミレーションという方法論がありますが，ここで紹介するConceptual Analyser(CA)はこの考え方に従ったものです．

5-3-2-1 実行環境

ここに上げたCAはProcide International社のWaltz Lisp(version 5.0)で記述されています．

Waltz LispはUNIX上で標準となっているFrantz Lispに準じた処理系で，PC-9800シリーズでもこの程度のプロセスならVAX 11／780と同程度のパフォーマンスを維持します．

Waltz Lispの起動は>WLSと入力します．プログラムのロードは，起動後に(load" ca.l)と入力します．

正常にロードできるとプロンプトが出ますので，例えば>(ca '(JOHN WENT HOME))と英文を入力すると処理が開始され，結果のCD表現が示されます．

5-3-2-2　CAの構成

CAはディクショナリの部分と処理部分とに分かれます．ディクショナリにはputprop関数によって各単語の属性が設定されます．

各単語は前後にある単語に依存するCD表現が記述されています．これはTEST：とACTION：によって構成されます．

TEST：はACTION：を実行するための条件部であり，一種のプロダクションシステムとなっています．TEST：部が成立したときに，それと対になったACTION：実行されます．

このディショナリは人間で言えば，長期記憶(LTM(LONGTERM MEMORY))にあたります．CAでは人間の短期記憶(STM(SHORTTERM MEMORY))に対応するメモリとしてC-LIST，R-LISTを持っています．

C-LISTは入力した単語によって引き起こされた概念をリストアップしておくConceptのListです．

R-LISTはC-LISTに存在する概念間の結び付き，C-LISTへの概念の追加を決定するルールが入っています．

5-3-2-3　CAの位置付け

CAにおける自然言語理解の方法は，従来の文法中心の方法とは逆に文法は言語の理解において従来思われていた程中心的ではなく，意味的情報を中心とした解析方法です．

自然言語処理の方法としては，Lexically-driven case-based parsingという方法になります．

これは入力がLexicon(単語)をキーに解析を進め，解析がCD表現という一種のフレームを生成する方法ということです．

自然言語処理の方法としては，他にATN(拡張遷移ネットワーク)やLexical Functional Grammer(LFG)等があります．

特に，LFGを用いた方法はICOT(新世代コンピュータ開発機構)やCMU(カーネギーメロン大学)のプロジェクトを始め，数多く研究されています．

ここではLFGは紹介しませんが，本プログラムCAをマスターされ，自然言語に興味を持たれた方はCAによるCD表現とLFGによるC-STRUCTURE，F-STRUCTUREという構造の融合を試みてください．

また，文脈の理解を中心にAIを考えたい方は，SAMと呼ばれるプログラムがInside Computer Understanding等に示されているので，プログラミングしてみるのも良いでしょう．

さらに，文の曖昧さを問題とするなら，文脈情報として文法だけでなく，多くの知識が必要になります．ここで自然言語理解のためには文の解析だけでなく知識ベースの構築，管理や信念システムといわれるシステムの研究が必要となります．

これらはLISPでもできますが，多くはPrologによって作られているケースが増えています．興味のある読者はLISP，Prologを使って，システムを構築してみるとよいでしょう．

プログラム 5-3-2 TECHCA.L

```
;;=======================================================================
;;
;;                      CONCEPTUAL ANALYZER
;;     Defined by Lawrence Birnbaum and Mallory Selfridge
;;     Original version implemented in T language by Hideto Tomabechi in 1985
;;     Converted to Waltz Lisp in 1986
;;
;;=======================================================================

;;============================================================
;; * EXSAMPLE SENTENCES *
;;============================================================
;;Example sentences are as follows:
;;
;;     (ca '(JOHN WENT HOME))
;;     (ca '(JOHN IS GOING TO TAKE AN AIRPLANE FROM TOKYO TO NEW YORK))
;;     (ca '(JOHN IS GOING TO NEW YORK))
;;     (ca '(JOHN TOOK THE AIRPLANE HOME FROM TOKYO))
;;     (ca '(JOHN TOOK A NEW BOOK TO NEW YORK FROM TOKYO ON AN AIRPLANE))
;;     (ca '(JOHN GAVE MARY A PUNCH))
;;     (ca '(JOHN GAVE MARY THE PUNCH))
;;
;;==========================================================
;; * CONCEPTUAL ANALYZER BODY *
;;==========================================================

(load 'CA.LIB)  ;; Loading primitive function definitions
(setq C-LIST nil)  ;;global variable to contain concepts
(setq R-LIST nil)  ;;global variable to contain requests
(setq T t)

;;
(def set-property (lambda (object attr value)
      (putprop object value attr]

;; * DICTIONARY *
;; Conceptual dictionary contains following concepts:
;;  JOHN  MARY  BOY  BOOK  DESK  ASPIRIN  BUS  AIRPLANE
;;  GIVE  GAVE  TAKE  TOOK  WENT  IS  GOING  A  AN THE
;;  TO  FROM  ON  BY  SICK  HAPPY  DEPRESSED  PUNCH
;;  KISS  HOME  YORK TOKYO NEW

(set-property 'JOHN
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list ;action
                              '(PP (CLASS (HUMAN)) (NAME (JOHN]

(set-property 'MARY
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list  ;action
                              '(PP (CLASS (HUMAN)) (NAME (MARY]

(set-property 'BOY
             'request
             '( ((TEST: T)
                 (ACTION: ( (add-c-list
                             '(PP (CLASS (HUMAN))
                                  (SEX (MALE))
                                  (AGE (YOUNG]
```

```
(set-property 'BOOK
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (PHYS-OBJ))
                                              (TYPE (BOOK]

(set-property 'DESK
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (PHYS-OBJ))
                                              (TYPE (DESK]

(set-property 'ASPIRIN
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (PHYS-OBJ))
                                          (TYPE (MEDICINE))
                                              (NAME (ASPIRIN]

(set-property 'BUS
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (PHYS-OBJ))
                                          (TYPE (VEHICLE))
                                              (NAME (BUS]

(set-property 'AIRPLANE
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (PHYS-OBJ))
                                          (TYPE (VEHICLE))
                                              (NAME (AIRPLANE]

(set-property 'GIVE
              'request
              '( ((TEST: T)  ; by default, GIVE is on ATRANS action
                  (ACTION: ( (add-c-list
                              '(ATRANS (ACTOR (nil))
                                       (OBJECT (nil))
                                       (TO (nil))
                                       (FROM (nil))
                                   (TIME (PRESENT))) ))
                 )
                )
                ((TEST: (precede '(PP (CLASS (HUMAN))) '(ATRANS)) )
                   ;;find a HUMAN on C-LIST preceding ATRANS structure?
                  (ACTION: ( (fill_slot
                              '(ACTOR FROM) ;;slots
                              (precede '(PP (CLASS (HUMAN))) '(ATRANS)) ;filler
                              '(ATRANS)))) ;;structure
                         ;;put it in the ACTOR and FROM slot of ATRANS
                )
                ((TEST: (follow '(PP (CLASS (HUMAN))) '(ATRANS)) )
                   ;;find a HUMAN on C-LIST following ATRANS structure?
                  (ACTION: ( (fill_slot
                              '(TO)
                              (follow '(PP (CLASS (HUMAN))) '(ATRANS))
                              '(ATRANS))) )
                         ;;Put it in the TO slot of the ATRANS
                )
                ((TEST: (follow '(PP (CLASS (PHYS-OBJ))) '(ATRANS)) )
                   ;;find a PHYS-OBJ on C-LIST following ATRANS structure?
```

```
                    (ACTION: ( (fill_slot
                                '(OBJECT)
                                (follow '(PP (CLASS (PHYS-OBJ))) '(ATRANS)) )
                                '(ATRANS]
                              ;;Put it in the OBJECT slot of the ATRANS

(set-property 'GAVE           ;;same as give, except it is past tense.
              'request
             '( ((TEST: T)
                 (ACTION: ( (add-c-list
                              '(ATRANS (ACTOR (nil))
                                       (OBJECT (nil))
                                       (TO (nil))
                                       (FROM (nil))
                                    (TIME (PAST))) ))
                 )
                )
               ((TEST: (precede '(PP (CLASS (HUMAN))) '(ATRANS)) )
                  ;;find a HUMAN on C-LIST preceding ATRANS structure?
                 (ACTION: ( (fill_slot
                              '(ACTOR FROM) ;;slots
                              (precede '(PP (CLASS (HUMAN))) '(ATRANS)) ;;filler
                               '(ATRANS)))) ;;structure
                            ;;put it in the ACTOR and FROM slot of ATRANS
               )
               ((TEST: (follow '(PP (CLASS (HUMAN))) '(ATRANS)) )
                  ;;find a HUMAN on C-LIST following ATRANS structure?
                 (ACTION: ( (fill_slot
                              '(TO)
                              (follow '(PP (CLASS (HUMAN))) '(ATRANS))
                              '(ATRANS))) )
                            ;;Put it it the TO slot of ATRANS
               )
               ((TEST: (follow '(PP (CLASS (PHYS-OBJ))) '(ATRANS)) )
                  ;;find a PHYS-OBJ on C-LIST following ATRANS structure?
                 (ACTION: ( (fill_slot
                              '(OBJECT)
                              (follow '(PP (CLASS (PHYS-OBJ))) '(ATRANS))
                              '(ATRANS]
                            ;;put it in the OBJECT slot of the ATRANS


(set-property 'TAKE
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list
                               '(PTRANS (ACTOR (nil))
                                        (OBJECT (nil))
                                        (TO (nil))
                                        (FROM (nil))
                                    (TIME (PRESENT))) ))
                  )
                 )
                    ;;If a MEDICINE consept comes right after TAKE, then
                    ;;TOOK is a INGEST action.
                ((TEST: (attached '(PTRANS)
                                  '(PP (CLASS (PHYS-OBJ))
                                       (TYPE (MEDICINE)))))
                    ;;find a MEDICINE on C-LIST attached PTRANS structure?
                  (ACTION: (  (add-c-list
                                 '(INGEST (ACTOR (nil))
                                          (OBJECT (nil))
                                      (TIME (PRESENT))))
                               (copy-filler '(PTRANS) '(INGEST) '(ACTOR))
```

```
                    (fill_slot
                     '(OBJECT)
                     (attached '(PTRANS)
                               '(PP (CLASS (PHYS-OBJ))
                                    (TYPE (MEDICINE))))
                     '(INGEST))
                    (del-c-list '(PTRANS)))
                   )
                 ;;put it in the object slot of the INGEST
   )
  ((TEST: (precede '(PP (CLASS (HUMAN))) '(PTRANS)) )
    ;;find a HUMAN on C-LIST preceding PTRANS structure?
   (ACTION: ( (fill_slot
                '(ACTOR OBJECT) ;;default for object is oneself
                  (precede '(PP (CLASS (HUMAN))) '(PTRANS)) ;;filler
                '(PTRANS))))    ;;structure
                 ;;put it in the ACTOR slot of PTRANS
  )
  ((TEST: (attached '(PTRANS)
                     '(PP (CLASS (PHY-OBJ))
                          (TYPE (VEHICLE)))))
      ;;find a VEHICLE on a C-LIST attached PTRANS structure?
   (ACTION: ( (fill_slot '(INST)
                         (attached
                               '(PTRANS)
                               '(PP (CLASS (PHYS-OBJ))
                                    (TYPE (VEHICLE))))
                         '(PTRANS))))
  )
  ((TEST: (OR (follow '(PP (CLASS (HUMAN))) '(PTRANS))
              (follow '(PP (CLASS (PHYS-OBJ))) '(PTRANS))) )
     ;;find HUMAN OR PHYS-OBJ on C-LIST following PTRANS?
   (ACTION: ( (fill_slot
                 '(OBJECT)
                  (follow '(PP) '(PTRANS))
                '(PTRANS))))
                 ;;Put it in the OBJECT slot of the PTRANS
  )
                 ;;JOHN TAKES MARY HOME type (but no conjugtions now)
   ((TEST: (follow '(PP (CLASS (LOCATION))) '(PTRANS)) )
        ;;find a LOCATION on C-LIST following PTRANS structure
   (ACTION: ( (fill_slot
                      '(TO)
                      (follow '(PP (CLASS (LOCATION))) '(PTRANS))
                      '(PTRANS))))
                 ;;put it in the TO slot of the PTRANS
  )
                 ;;JOHN TAKES THE BOOK ON THE DESK type
  ((TEST: (and (follow '(PREP (POS)) '(PTRANS))
                (attached '(PREP (POS)) '(PP) )) )
     ;;find POSition prep. on C-LIST following PTRANS structure?
     ;;and some comcept attached to position preposition token?
   (ACTION: ( (fill_slot
                  '(FROM)
                  (attached '(PREP (POS)) '(PP) )
                  '(PTRANS))
                 ;;put attached word in the FROM slot of the PTRANS
                (del-c-list '(PREP (POS)))
                (add-c-list
                   '(GRASP (ACTOR (nil))
                            (OBJECT (nil))
                          (TIME (PRESENT))))
                (copy-filler '(PTRANS) '(GRASP) '(ACTOR OBJECT))
```

```
                    (fill_slot
                     '(INST)
                       (findp '(GRASP) C-LIST 0)
                     '(PTRANS))
                    (del-c-list '(GRASP]


(set-property 'TOOK       ;;TOOK is same as TAKE but is PAST
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list
                               '(PTRANS (ACTOR (nil))
                                        (OBJECT (nil))
                                        (TO (nil))
                                        (FROM (nil))
                                        (TIME (PAST))) ))
                  )
                 )
                 ((TEST: (attached '(PTRANS)
                                   '(PP (CLASS (PHYS-OBJ))
                                        (TYPE (MEDICINE)))))
                   ;;find a MEDICINE on C-LIST attached PTRANS structure
                  (ACTION: (  (add-c-list
                                '(INGEST (ACTOR (nil))
                                         (OBJECT (nil))
                                         (TIME (PAST))))
                              (copy-filler '(PTRANS) '(INGEST) '(ACTOR))
                              (fill_slot
                                '(OBJECT)
                                 (attached '(PTRANS)
                                           '(PP (CLASS (PHYS-OBJ))
                                                (TYPE (MEDICINE))))
                                 '(INGEST))
                               (del-c-list '(PTRANS)))
                          )
                 ;;put it in the object slot of the INGEST
                 )
                 ((TEST: (precede '(PP (CLASS (HUMAN))) '(PTRANS)) )
                   ;;find a HUMAN on C-LIST preceding PTRANS structure?
                  (ACTION: ( (fill_slot
                               '(ACTOR OBJECT) ;;default for object is oneself
                              (precede '(PP (CLASS (HUMAN))) '(PTRANS)) ;;filler
                               '(PTRANS)))) ;;structure
                          ;;put it in the ACTOR slot of PTRANS
                 )
                 ((TEST: (attached '(PTRANS)
                                   '(PP (CLASS (PHYS-OBJ))
                                        (TYPE (VEHICLE)))))
                   ;;find a VEHICLE on C-LIST attached PTRANS structure?
                  (ACTION: ( (fill_slot (INST)       ;;slot
                                        (attached
                                          '(PTRANS)
                                          '(PP (CLASS (PHYS-OBJ))
                                               (TYPE (VEHICLE))))
                                        '(PTRANS))))
                 )
                 ((TEST: (OR (follow '(PP (CLASS (HUMAN))) '(PTRANS))
                             (follow '(PP (CLASS (PHYS-OBJ))) '(PTRANS))) )
                   ;;find HUMAN OR PHYS-OBJ on C-LIST following PTRANS
                  (ACTION: ( (fill_slot
                              '(OBJECT)
                              (follow '(PP) '(PTRANS))
                              '(PTRANS))))
```

```
                ;;Put it in the OBJECT slot of the PTRANS
          )
                ;;JOHN TOOK MARY HOME type
          ((TEST: (follow '(PP (CLASS (LOCATION))) '(PTRANS)) )
            ;;find a LOCATION on C-LIST following PTRANS structure?
           (ACTION: ( (fill_slot
                       '(TO)
                       (follow '(PP (CLASS (LOCATION))) '(PTRANS))
                       '(PTRANS))))
                ;;put it in the TO slot of the PTRANS
          )
                ;;JOHN TOOK THE BOOK ON THE DESK type
          ((TEST: (and (follow '(PREP (POS)) '(PTRANS))
                       (attached '(PREP (POS)) '(PP) )) )
           ;;find POSisiton prep. on C-LIST following PTRANS structure?
            ;;and some concept attached to position preposition token?
           (ACTION: ( (fill_slot
                         '(FROM)
                         (attached '(PREP (POS)) '(PP) )
                         '(PTRANS))
                ;;put attached word in the FROM slot of the PTRANS
                       (del-c-list '(PREP (POS)))
                       (add-c-list
                         '(GRASP (ACTOR (nil))
                                 (OBJECT (nil))
                            (TIME (PAST))))
                       (copy-filler '(PTRANS) '(GRASP) '(ACTOR OBJECT))
                       (fill_slot
                        '(INST)
                         (findp '(GRASP) C-LIST 0)
                        '(PTRANS))
                       (del-c-list '(GRASP]


(set-property 'WENT  ;;WENT is PTRANS. By default, ACTOR and OBJECT are same.
        'request
        '( ((TEST: T)
             (ACTION: ( (add-c-list
                         '(PTRANS (ACTOR (nil))
                                  (OBJECT (nil))
                                  (TO (nil))
                                  (FROM (nil))
                             (TIME (PAST))) ))
             )
           )
           ((TEST: (precede '(PP (CLASS (HUMAN))) '(PTRANS)) )
             ;;find a HUMAN on C-LIST preceding PTRANS structure?
            (ACTION: ( (fill_slot
                        '(ACTOR OBJECT) ;;default for object is oneself
                        (precede '(PP (CLASS (HUMAN))) '(PTRANS)) ;;filler
                          '(PTRANS))))    ;;structure
                ;;put it in the ACTOR OBJECT slot of PTRANS
                   )
           ((TEST: (follow '(PP (CLASS (LOCATION))) '(PTRANS)) )
           (ACTION: ( (fill_slot
                         '(TO)
                         (follow '(PP (CLASS (LOCATION))) '(PTRANS))
                         '(PTRANS]


(set-property 'IS   ;;Example for BE
        'request
                ;;
```

```
'( ((TEST: (precede '(PP (CALSS (HUMAN))) '(STATE)) )
      ;;find a HUMAN on C-LIST preceding STATE structure?
     (ACTION: ( (fill_slot
                  '(ACTOR) ;; slot
                   (precede '(PP (CLASS (HUMAN))) '(STATE))
                  '(STATE)))) ;;structure
                ;;put it in the ACTOR slot of STATE
   )
   ((TEST: T)
    (ACTION: ( (add-c-list
                '(STATE (ACTOR (nil))
                        (OBJECT (nil))
                        (TIME (PRESENT))) ))
    )
   )
   ((TEST: (follow '(AC) '(STATE)) )
     ;;find a Abstract Concept on C-LIST following STATE?
     (ACTION: (fill_slot
               '(OBJECT)
               (follow '(AC) '(STATE))  ;;Fill with AC
               '(STATE)))
              ;;put it in the OBJECT slot of the STATE
   )
   ((TEST: (attached '(STATE) '(PP)) )
     ;;find a Picture Producer on C-LIST attached to IS?
    (ACTION: ( (add-c-list
                 '(ISA (ACTOR (nil))
                       (OBJECT (nil))
                   (TIME (PRESENT))))
               (copy-filler '(STATE) ;;copy subject from STATE
                            '(ISA)
                            '(ACTOR))
               (fill_slot
                '(OBJECT)  ;;PP attached to IS is the object
                (attached '(STATE) '(PP))
                '(ISA))
               (del-c-list '(STATE]
;;
;;
;;
;;
(set-property 'GOING
              'request
              '( ((TEST: T)  ;;delete STATE created by preceding IS
                   (ACTION: ( (del-c-list '(STATE)) )))
                             ;;LOCATION found in the C-LIST?
                 ((TEST: (findp '(PP (CLASS (LOCATION))) C-LIST 0))
                  (ACTION: ( (add-c-list    ;;add PTRANS structure
                               '(PTRANS (ACTOR (nil))
                                        (OBJECT (nil))
                                        (TO (nil))
                                 (TIME (PRESENT))) )
                             (fill_slot  ;;JOHN IS GOING HOME type default
                             '(TO)
                              (findp '(PP (CLASS (LOCATION))) C-LIST 0)
                             '(PTRANS))
                            )
                  )
                 )
                 ((TEST: (precede '(PP (CLASS (HUMAN))) '(PTRANS)) )
                   ;;find a HUMAN on C-LIST preceding PTRANS structure?
                  (ACTION: ( (fill_slot
                             '(ACTOR OBJECT) ;;default for object is oneself
```

```
                    (precede '(PP (CLASS (HUMAN))) '(PTRANS)) ;;filler
                    '(PTRANS)))) ;;structure
                  ;;put it in the ACTOR OBJECT slot of PTRANS
        )
                  ;;Any action found?
        ((TEST: (OR (findp '(ATRANS) C-LIST 0)
                    (findp '(PTRANS) C-LIST 0)
                    (findp '(MTRANS) C-LIST 0)
                    (findp '(SPEAK) C-LIST 0)
                    (findp '(INGEST) C-LIST 0)
                    (findp '(MBUILD) C-LIST 0)
                    (findp '(GRASP) C-LIST 0)
                    (findp '(EXPEL) C-LIST 0)
                    (findp '(PROPEL) C-LIST 0)
                    (findp '(ATTEND) C-LIST 0)
                    (findp '(MOVE) C-LIST 0) ) )
        (ACTION: ((OR
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(ATRANS))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(PTRANS))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(MTRANS))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(SPEAK))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(INGEST))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(MBUILD))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(GRASP))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(EXPEL))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(PROPEL))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(ATTEND))
                     (fill_slot '(TIME) '(CPT (FUTURE)) '(MOVE))
                   (del-c-list '(STATE)) ;;delete STATE created by IS
                   (del-c-list '(PREP (DIR] ;;delete TO after GOING

;;
;;
;;
;;
;;
(set-property 'A
          'request
          '( ((TEST: T)
               (ACTION: ( (add-c-list '(INDEF (REF)))))
             )
             ((TEST: (and (follow '?X '(INDEF (REF)))
                          (not (follow '(AC (CLASS (STATE)))
                                       '(INDEF (REF))) ))
                                  ;;any concept following except STATE?
              )
               (ACTION: ( (ADD_fill_slot 'INDEF    ;;slot
                                       '(REF)     ;;filler
                                         (follow '?X '(INDEF (REF]
                                                  ;;structure

(set-property 'AN
          'request
          '( ((TEST: T)
               (ACTION: ( (add-c-list '(INDEF (REF)))))
             )
             ((TEST: (and (follow '?X '(INDEF (REF)))
                          (not (follow '(AC (CLASS (STATE)))

                                       '(INDEF (REF))) ))
                                  ;;any concept following except STATE?
              )
               (ACTION: ( (ADD_fill_slot 'INDEF      ;;slot
```

```
                                '(REF) ;;filler
                                (follow '?X '(INDEF (REF]
                                         ;;structure


 ;;THE is same as A and AN. except that it adds (DEF (REF)).
(set-property 'THE
             'request
             '( ((TEST: T)
                 (ACTION: ( (add-c-list '(DEF (REF)))))
                )
                ((TEST: (and (follow '?X '(DEF (REF)))
                             (not (follow '(AC (CLASS (STATE)))
                                          '(DEF (REF))) ))
                                  ;;any concept following except STATE?
                 )
                 (ACTION: ( (ADD_fill_slot 'DEF       ;;slot
                                '(REF)       ;;filler
                                (follow '?X '(DEF (REF]
                                        ;;structure


(set-property 'TO        ;;(PREP (DIR)) is a concept denoting a token for
             'request    ;;some direction. (PREposition DIRection)
             '( ((TEST: T)
                 (ACTION: ( (add-c-list '(PREP (DIR)))))
                )
                ((TEST: (attached '(PREP (DIR))  ;;any LOCATION attached?
                                  '(PP (CLASS (LOCATION)))))
                 (ACTION: ( (fill_slot
                              '(TO)         ;;slot
                              (attached '(PREP (DIR))
                                        '(PP (CLASS (LOCATION))))
                              '(PTRANS))
                            (del-c-list '(PREP (DIR]

(set-property 'FROM
              'request
              '( ((TEST: (findp '(PTRANS) C-LIST 0))
                          (ACTION: ( (add-c-list '(PREP (DIR)))))
               )
               ((TEST: (follow '?X '(PREP (DIR))))  ;;any concept following?
                (ACTION: ( (fill_slot
                             '(FROM)          ;;slot
                             (follow '?X '(PREP (DIR)))  ;;filler
                             '(PTRANS))
                           (del-c-list '(PREP (DIR]


  ;;
  ;;
(set-property 'BY
              'request
             '( ((TEST: (findp '(PTRANS) C-LIST 0))
                 (ACTION: ( (add-c-list '(PREP (INST)))))
                )
                ((TEST: (attached '(PREP (INST)) ;;filler
                                  '(PP (CLASS (PHYS-OBJ))
                                       (TYPE (VEHICLE)))))
                 (ACTION: ( (fill_slot '(INST)
                             (attached '(PREP (INST)) ;;filer
                                       '(PP (CLASS (PHYS-OBJ))
                                            (TYPE (VEHICLE))))
                             '(PTRANS))  ;;structure
```

```
                    (del-c-list '(PREP (INST]


(set-property 'SICK        ;;A bit generalized sickness.
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list
                                 '(AC (CLASS (STATE))
                                      (TYPE (PHYSICAL))
                                          (VALUE (-7]


(set-property 'HAPPY
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list
                                 '(AC (CLASS (STATE))
                                      (TYPE (MENTAL))
                                          (VALUE (+10]


(set-property 'DEPRESSED
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list
                                 '(AC (CLASS (STATE))
                                      (TYPE (MENTAL))
                                          (VALUE (-8]


  ;;
  ;;
  ;;
  ;;and finally it delete the ATRANS structure from the C-LIST.
(set-property 'PUNCH
              'request         ;;'A' currently in the C-LIST
             '( ((TEST: (findp '(INDEF (REF)) C-LIST 0))
                  (ACTION: ( (del-c-list (INDEF (REF)))
                             (add-c-list '(PROPEL (ACTOR (nil))
                                                  (OBJECT (FIST))
                                                        (TO (nil))
                                                        (FROM (nil))))))
                 )
                 ((TEST: (precede '(ATRANS) '(PROPEL)) )
                  (ACTION: ( (copy-filler '(ATRANS) '(PROPEL) '(ACTOR FROM))
                             (copy-filler '(ATRANS) '(PROPEL) '(TO))
                             (del-c-list '(ATRANS))))
                 )
                 ((TEST: (precede '(PTRANS) '(PROPEL)) )
                  (ACTION: ( (move-filler
                               '(PTRANS) 'ACTOR '(PROPEL) '(TO))
                            (MOVE-FIILER
                               '(PTRANS) 'FROM '(PROPEL) '(ACTOR FROM))
                                     ;;when FROM comes before PUNCH
                             (del-c-list '(PTRANS))))
                 )
                 ((TEST: (follow '(PP (CLASS (HUMAN))) '(PROPEL)) )
                  ;;find a HUMAN on C-LIST following PROPEL structure?
                  (ACTION: ( (fill_slot
                               '(ACTOR FROM)   ;;slots
                              (follow '(PP (CLASS (HUMAN))) '(PROPEL)) ;;filler
                              '(PROPEL))))     ;;structure
```

```
                    ;;put it in the ACTOR and FROM slot of PROPEL
          )
           ((TEST: (AND (findp '(DEF (REF)) C-LIST 0)
                        (findp '(ATRANS) C-LIST 0)))
            (ACTION: ( (add-c-list
                           '(PTRANS (ACTOR (nil))
                                 (OBJECT (PP (CLASS (PHYS-OBJ))
                                             (TYPE (BEVERAGE))
                                              (NAME (PUNCH))
                                              (REF (DEF))))
                                         (FROM (nil))
                                         (TO (nil))))
                       (del-c-list '(DEF (REF))) ))
           )
           ((TEST: (precede '(ATRANS) '(PTRANS)) )
            (ACTION: ( (copy-filler '(ATRANS) '(PTRANS) '(ACTOR FROM))
                       (copy-filler '(ATRANS) '(PTRANS) '(TO))
                       (del-c-list '(ATRANS]

 ;;
 ;;
 ;;
(set-property 'KISS
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(MOVE (ACTOR (nil))
                                                (OBJECT (LIPS))
                                                      (TO (nil))))))
           )
           ((TEST: (precede '(ATRANS) '(MOVE)) )
            (ACTION: ( (copy-filler '(ATRANS) '(MOVE) '(ACTOR))
                       (copy-filler '(ATRANS) '(MOVE) '(TO))
                       (del-c-list '(INDEF (REF)))
                       (del-c-list '(ATRANS]


(set-property 'HOME          ;;(NAME (HOME)) is bit strange but at least good
              'request       ;;enough to distinguish from other LOCATIONs
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (LOCATION))
                                              (NAME (HOME]


(set-property 'YORK          ;;This is added to process NEW YORK.
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (LOCATION))
                                              (TYPE (CITY))
                                              (NAME (YORK]


(set-property 'TOKYO         ;;Perhaps the world's most exciting city.
              'request
              '( ((TEST: T)
                  (ACTION: ( (add-c-list '(PP (CLASS (LOCATION))
                                          (TYPE (CITY))
                                              (NAME (TOKYO]


(set-property 'NEW           ;;To handle NEW BOOK type and NEW YORK
```

```
            'request
            '( ((TEST: T)    ;;The thing is new.
                (ACTION: ( (add-c-list
                             '(AC (CLASS (PHYS-ST))
                                  (TYPE (NEWLINESS))
                                  (VALUE (+10))))))
               )
                             ;;Is PHYS-OBJ attached to NEW?


               ((TEST: (attached '(AC (CLASS (PHYS-ST))
                                      (TYPE (NEWLINESS))
                                      (VALUE (+10)))
                                   '(PP (CLASS (PHYS-OBJ)))))
                 (ACTION: ( (ADD_fill_slot 'STATE      ;;slot
                                           '(NEW)       ;;filler
                                           (attached '(AC (CLASS (PHYS-ST))
                                                          (TYPE (NEWLINESS))
                                                          (VALUE (+10)))
                                                      '(PP (CLASS (PHYS-OBJ)))))
                                                       ;;structure
                            (del-c-list '(AC (CLASS (PHYS-ST))
                                             (TYPE (NEWLINESS))
                                             (VALUE (+10)))))
                           )
               )
                              ;;If YORK is attached to NEW
               ((TEST: (attached '(AC (CLASS (PHYS-ST))
                                      (TYPE (NEWLINESS))
                                      (VALUE (+10))))
                                  '(PP (CLASS (LOCATION))
                                     (TYPE (CITY))
                                     (NAME (YORK))) ) )
                               ;;it is the name of a city called NEW YORK
                 (ACTION: ( (add-c-list '(PP (CLASS (LOCATION))
                                             (TYPE (CITY))
                                             (NAME (NEW YORK))))
                            (del-c-list '(PP (CLASS (LOCATION))
                                             (TYPE (CITY))
                                             (NAME (YORK))))
                            (del-c-list '(AC (CLASS (PHYS-ST))
                                             (TYPE (NEWLINESS))
                                             (VALUE (+10]


  ;;============================================================
  ;; * CONTROL STRUCTURE *
  ;;============================================================
    ;;
    ;;
    ;;
    ;;

(def ca (lambda (input)
    (clear_globals)
      (terpr)
    (msg "CONCEPTUAL ANALYSES WILL BE TRIED ON: ") (terpr) (msg input)
      (terpr)
    (process_next_input input]
```

```
  ;;
  ;;
(def process_next_input (lambda (list)
   (cond ((null list) (end))
         (t (msg t t "READ THE NEXT WORD: " (car list) )
                (process  (car list))
                 (process_next_input (cdr list]

  ;;
  ;;
(def end (lambda ()
   (msg t "***ALL WORDS PROCESSED***" t)
   (print_result C-LIST)
   (reset]

  ;;print_result prints the final result in the C-LIST
(def print_result (lambda (cpt-list)
     (prog (f-res)
  (cond ((null cpt-list) nil)
        (t (msg "FINAL RESULT: ") (terpr)
         (setq f-res (get 'MEMORY (car cpt-list))) (pp f-res)
              (print_result (cdr cpt-list]

  ;;clear two global variables.
(def clear_globals (lambda ()
    (setq C-LIST nil)
    (setq R-LIST nil)
    (setq CONNUM 0)
    (setq REQNUM 0]
  ;;
(def load_request (lambda (word)
      (let ((entry (get word 'request)))
                 (cond ((null entry) (msg t "NO ENTRY FOUND FOR " word t))
                       (t (msg t "LOAD THE REQUESTS FROM THE DICTIONARY"
                          t "The entry for " word " is: "
                          t ) (pp entry))
                 (cond ((null (cdr entry)) (push_requests entry))
                       (t (push_requests (reverse entry]

  ;;
(def push_requests (lambda (entry)
      (prog (req-id)
    (cond ((null entry) nil)
          (t (let ((req (generate-symbol 'REQ)))
                  (set-property 'MEMORY req (car entry))
                  (msg t t  req " IS REQUEST IDENTIFER FOR:") (terpr)
                         (setq req-id (car entry))
                          (pp req-id)
                   (push R-LIST req)
                   (push_requests (cdr entry]
  ;;
  ;;
(def push (macro (l)
      (list 'car (list 'setq (cadr l) (list 'cons (caddr l)) (cadr l]

  ;;
(def add-c-list (lambda (item)
    (let ((cpt (generate-symbol 'CON)))
          (set-property 'MEMORY cpt item)
          (push C-LIST cpt)
          (msg t "A NEW CONCEPT ADDED TO THE C-LIST: "
              t "  " cpt " is assigned for " t ) (pp item) (terpr]
```

```
(def generate-symbol (lambda (tag)
                (cond ((eq 'REQ tag) (progn
                                       (setq REQNUM (add1 REQNUM))
                                       (implode
                                       (append (explode tag) (explode REQNUM)))))
                      ((eq 'CON tag) (progn
                                       (setq CONNUM (add1 CONNUM))
                                       (implode
                                       (append (explode tag) (explode CONNUM)))))
                          (t nil]


  ;;process is the main control structure for the conceptual analyzer.
(def process (lambda (atom)
    (load_request atom)
    (consider_request]

  ;;consider_request sends control to consider_request_body to attain
  ;;recursion
(def consider_request (lambda ()
    (consider_request_body R-LIST]

  ;;
  ;;
  ;;
(def consider_request_body (lambda (requests)
    (msg t "CONSIDERING THE ACTIVE REQUESTS...")
    (cond ((null requests) (msg t " No more request" t))
          (t (progn (setq request (car requests)) ;;assign sequentially
                    (setq test (cadar (get 'MEMORY request))) ;TEST body
                    (setq action (cadadr (get 'MEMORY request))) ;ACTION body
                      (msg t "CONSIDERING " request ":")
                      (cond ((do_test test)
                              (progn (do_action action)
                                     (setq R-LIST (delete request R-LIST))
                                     (consider_request_body (cdr requests))))
                            (t (consider_request_body (cdr requests]


  ;;
  ;;
(def evaluate (lambda (expr)
     (eval expr]

  ;;do_test simply evaluates TEST: body
(def do_test (lambda (test)
    (msg t "TESTING => " ) (pp test)
    (evaluate test]

  ;;
  ;;
(def do_action (lambda (action)
  (cond ((null action) nil)
        (t (progn (msg t "ACTION ACTIVATED:") (terpr) (pp action) (terpr)
              (evaluate (car action))
              (do_action (cdr action]

(def delete (lambda (exp list)   ; delete a element from a list
    (cond ((null list) nil)
          ((eq exp (car list)) (cdr list))
          (t (cons (car list) (delete exp (cdr list]
  ;;
  ;;
```

```
(def del-c-list (lambda (cpt)
   (setq C-LIST (delete (car (findp cpt C-LIST 0)) C-LIST]
;;=====================================================================
;; * TEST AND ACTION HANDLING FUNCTIONS *
;;=====================================================================
;;
  ;;
  ;;
(def findp (lambda (exp cpt-list position)
   (let ((concept (get 'MEMORY (car cpt-list))))
       (cond ((null cpt-list) nil)
             ((match exp concept nil)
              (list (car cpt-list) concept position))
            (t (findp exp (cdr cpt-list) (add1 position]


  ;;
  ;;
(def precede (lambda (first second)
   (progn (setq f-find (findp first C-LIST 0))
           (setq f-position (caddr f-find))
          (setq s-position (caddr (findp second C-LIST 0)))
         (cond ((and f-position s-position (greaterp f-position s-position))
                (list (car f-find) (cadr f-find))) ;; (car f-find) = CON.##
               (t nil]                          ;; (cadr f-find) = content


   ;;
   ;;
(def follow (lambda (first second)
   (progn (setq f-find (findp first C-LIST 0))
           (setq f-position (caddr f-find))
          (setq s-position (caddr (findp second C-LIST 0)))
         (cond ((and f-position s-position (lessp f-position s-position))
                (list (car f-find) (cadr f-find))) ;;
               (t nil]                          ;;

   ;;
   ;;
   ;;
(def attached (lambda (first second)
   (progn (setq f-position (caddr (findp first C-LIST 0)))
          (setq s-find (findp second C-LIST 0))
           (setq s-position (caddr s-find))
          (cond ((and f-position
                      s-position
                       (and (lessp (sub f-position s-position) 3)
                            (plusp (sub f-position s-position))))
                (list (car s-find) (cadr s-find))) ;;
                (t nil]                         ;;

   ;;
   ;;
   ;;
(def fill_slot (lambda (roles filler cd-head)
 (let ((slot-cpt (findp cd-head C-LIST 0))) ;;findp = cd form
  (cond ((null slot-cpt) nil)
       ((null roles) (setq C-LIST (delete (car filler) C-LIST)) )
       (t (set-property 'MEMORY
                        (car slot-cpt)
                        (setrole (car roles)
                                 (cadr filler)
                                 (cadr slot-cpt)))
```

```
            (fill_slot (cdr roles) filler cd-head]

  ;;
  ;;
  ;;
(def add_fill_slot (lambda (slot filler slot-cpt)
    (set-property 'MEMORY
                  (car slot-cpt)
                  (setrole slot filler (cadr slot-cpt)))
    (setq C-LIST (delete (car (findp (list slot) C-LIST 0)) C-LIST]

  ;;
  ;;
  ;;
(def copy-filler (lambda (old-cd new-cd roles)
  (cond ((null roles) nil)
        (t
           (let ((old (findp old-cd C-LIST 0))
                 (new (findp new-cd C-LIST 0)))
                (set-property 'MEMORY
                              (car new)
                              (setrole (car roles)
                              (get_filler_of_role_in_cd (car roles) (cadr old))
                                        (cadr new)))
                 (copy-filler old-cd new-cd (cdr roles]

  ;;
  ;;
  ;;
(def move-filler (lambda (old-cd old-role new-cd new-roles)
  (cond ((null new-roles) nil)
        (t
           (let ((old (findp old-cd C-LIST 0))
                 (new (findp new-cd C-LIST 0)))
                (set-property 'MEMORY
                              (car new)
                              (setrole (car new-roles)
                                (get_filler_of_role_in_cd old-role (cadr old))
                                        (cadr new)))
                 (move-filler old-cd old-role new-cd (cdr new-roles]

;;==================================================================
;; CONCEPTUAL ANALYZER
;;      1985, 1986
;;==================================================================
;;
;;*EOF*
```

プログラム 5-3-3　TECHCA.LIB

```
; ca.lib                     Waltz lisp version
;
;
; defined Macros-> UCI: loop, for, let, push, pop, cons-end.
; defined Func. -> MAC: consp, defprop.
; defined CD     -> CD : get_header_cd, get_role_list_cd, get_role_of_pair,
;                       get_filler_of_pair,
;                       get_filler_of_role_in_cd, setrole, ?, is-var, name:var,
;                       match, instantiate.
```

```
;
;
(def loop (macro (l)
  (loop1 (cdr l)
      (get_keyword 'initial l)
      (get_keyword 'result l]

(def loop1 (lambda (clauses i-body r-body)
     (append (list 'prog (var-list i-body))
          (setq_steps i-body)
            (cons 'loop (apply 'append (mapcar 'do-clause clauses)))
            (list '(go loop) 'exit (cons 'return r-body]


;
(def get_keyword (lambda (key l)
  (cdr (assoc key (cdr l]


;
(def do-clause (lambda (clause)
  (selectq (car clause)
      ((initial result) nil)
       (while (list (list 'or (cadr clause) '(go exit))))
       (do (cdr clause))
       (until (list (list 'and (cadr clause) '(go exit))))
       (msg "unknown keyword" clause]


;
(def var-list (lambda (l)
  (cond ((null l) nil)
      (t (cons (car l) (var-list (cddr l]

;setq_steps: (v1 e1 v2 e2...) => ((setq v1 e1) (setq v2 e2)...)
;
(def setq_steps (lambda (l)
  (cond ((null l) nil)
      ((null (cadr l)) (setq_steps (cddr l)))
      (t (cons (list 'setq (car l) (cadr l))
               (setq_steps (cddr l]

;
(def for (macro (l)
  (for1 (cadr l)
      (get_keyword 'when l)
      (get_keyword 'do l)
      (get_keyword 'save l)
      (get_keyword 'exists l]

(def for1 (lambda (in when do save exists)
  (cons (for-mapfn when do save exists)
      (cons (for-lambda (car in) when do save exists)
            (cddr in]

(def for-mapfn (lambda (when do save exists)
  (cond (do 'mapc)
      (exists 'some)
      (when 'mapcan)
      (t 'mapcar]

(def for-lambda (lambda (var when do save exists)
  (list 'function
      (cons 'lambda
```

```
             (cons (list var)
                   (cond (when (for-when when do save))
                         (t (or do save exists]

(def for-when (lambda (when do save)
  (list (list 'cond
             (append (add-progn when)
                     (or do (list (cons 'ncons (add-progn save]

(def some (lambda (func list)
    (cond ((equal nil list) nil)
          ((funcall func (car list)) list)
           (t (some func (cdr list]

(def add-progn (lambda (l)
  (cond ((cdr l) (list (cons 'progn l)))
      (t l]

;
(def let* (macro (l)
  (let1 (reverse (cadr l)) nil nil (cddr l]

(def let1 (lambda (l vars vals body)
  (cond ((null l) (cons (cons 'lambda (cons vars body)) vals))
      (t (let1 (cddr l)
               (cons (cadr l) vars)
               (cons (car l) vals)
               body]
;
;(def push (macro (l)
;  (list 'car (list 'setq (caddr l) (list 'cons (cadr l) (caddr l]

(def pop (macro (l)
  (list 'prog1
      (list 'car (cadr l))
      (list 'setq (cadr l) (list 'cdr (cadr l]

(def cons-end (lambda (l x) (append l (list x]

(def defprop
 (nlambda ($$$exp$$)
      (putprop (car $$$exp$$) (cadr $$$exp$$) (caddr $$$exp$$))))


;Definition of CD Functions
;
;
(def get_header_cd (lambda (x) (car x))]
(def get_role_list_cd (lambda (x) (cdr x))]

;
(def get_role_of_pair (lambda (x) (car x))]
(def get_filler_of_pair (lambda (x) (cadr x))]

;
(def get_filler_of_role_in_cd (lambda (role cd)
  (let* (pair (assoc role (get_role_list_cd cd)))
      (and pair (get_filler_of_pair pair]

;
(def setrole (lambda (role filler cd)
  (cons (get_header_cd cd)
        (append (for (pair in (get_role_list_cd cd))
```

```
                (when (not (equal (get_role_of_pair pair) role)))
                (save pair))
              (list (list role filler]

;
(dfatc |?| 2 (lambda () (list '*var* (read t]

(def is-var (lambda (x) (and (consp x) (equal (car x) '*var*]

(def is-interrogative-var (lambda (x) (and (consp x) (equal (car x) '***]

(def name:var (lambda (x) (and (consp x) (consp (cdr x)) (cadr x]

(def consp (lambda (x) (cond ((not (atom x)) x]

;
(def match (lambda (pat const bindings)
  (let* (binding-form (or bindings (list t)))
    (cond ((or (null const) (equal pat const)) binding-form)
        ((is-var pat) (match-var pat const binding-form))
        ((or (atom const) (atom pat)) nil)
        ((equal (get_header_cd pat) (get_header_cd const))
         (match-args (get_role_list_cd pat) const binding-form]

;
(def match-args (lambda (pat-args const binding-form)
  (loop (initial pat-arg nil const-val nil)
      (while (setq pat-arg (pop pat-args)))
      (do (setq const-val (get_filler_of_role_in_cd (get_role_of_pair pat-arg) const)))
      (while (setq binding-form
                   (match (get_filler_of_pair pat-arg)
                          const-val
                          binding-form)))
      (result binding-form]

;
(def match-var (lambda (pat const binding-form)
  (let* (var-value (get_filler_of_role_in_cd (name:var pat) binding-form))
    (cond (var-value (match var-value const binding-form))
        (t (cons-end binding-form (list (name:var pat) const]

;
(def instantiate (lambda (cd-form bindings)
  (cond ((atom cd-form) cd-form)
      ((is-var cd-form)
       (instantiate (get_filler_of_role_in_cd (name:var cd-form) bindings)
                    bindings))
      (t (cons (get_header_cd cd-form)
               (for (pair in (get_role_list_cd cd-form))
                    (save (list (get_role_of_pair pair)
                                (instantiate (get_filler_of_pair pair)
                                             bindings]
```

```
A>WLS

Waltz Lisp 5.12, (C) 1986 by ProCode Intl.

-> ( LOAD "TECHCA.L" )
@ t
-> ( LOAD "TECHCA.LIB" )
@ t
-> ( CA '(JOHN WENT HOME))

CONCEPTUAL ANALYSES WILL BE TRIED ON:
(JOHN WENT HOME)


READ THE NEXT WORD: JOHN
LOAD THE REQUESTS FROM THE DICTIONARY
The entry for JOHN is:
(setq entry
'(((TEST: t) (ACTION: ((add-c-list '(pp (CLASS (HUMAN)) (NAME (JOHN))))))))


req1 IS REQUEST IDENTIFER FOR:
(setq req-id
'((TEST: t) (ACTION: ((add-c-list '(pp (CLASS (HUMAN)) (NAME (JOHN))))))))

CONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req1:
TESTING => (setq test
't)

ACTION ACTIVATED:
(setq action
'((add-c-list '(pp (CLASS (HUMAN)) (NAME (JOHN))))))


A NEW CONCEPT ADDED TO THE C-LIST:
  CON1 is assigned for
(setq item
'(pp (CLASS (HUMAN)) (NAME (JOHN))))


CONSIDERING THE ACTIVE REQUESTS...
 No more request


READ THE NEXT WORD: WENT
LOAD THE REQUESTS FROM THE DICTIONARY
The entry for WENT is:
(setq entry
'(((TEST: t)
   (ACTION:
    ((add-c-list
      (quote
       (PTRANS (ACTOR (nil))
               (object (nil))
               (TO (nil))
               (FROM (nil))
               (TIME (PAST))
       )
      )
     )
    )
   )
  )
  ((TEST: (precede '(pp (CLASS (HUMAN))) '(PTRANS)))
   (ACTION:
    ((fill_slot '(ACTOR object)
                (precede '(pp (CLASS (HUMAN))) '(PTRANS))
                '(PTRANS)
     )
    )
   )
```

```
  )
  ((TEST: (follow '(pp (CLASS (LOCATION))) '(PTRANS)))
   (ACTION:
    ((fill_slot '(TO) (follow '(pp (CLASS (LOCATION))) '(PTRANS)) '(PTRANS)))
   )
  )
 ))


req2 IS REQUEST IDENTIFER FOR:
(setq req-id
'((TEST: (follow '(pp (CLASS (LOCATION))) '(PTRANS)))
  (ACTION:
   ((fill_slot '(TO) (follow '(pp (CLASS (LOCATION))) '(PTRANS)) '(PTRANS)))
  )
 ))


req3 IS REQUEST IDENTIFER FOR:
(setq req-id
'((TEST: (precede '(pp (CLASS (HUMAN))) '(PTRANS)))
  (ACTION:
   ((fill_slot '(ACTOR object)
               (precede '(pp (CLASS (HUMAN))) '(PTRANS))
               '(PTRANS)
    )
   )
  )
 ))


req4 IS REQUEST IDENTIFER FOR:
(setq req-id
'((TEST: t)
  (ACTION:
   ((add-c-list
     (quote
      (PTRANS (ACTOR (nil))
              (object (nil))
              (TO (nil))
              (FROM (nil))
              (TIME (PAST))
      )
     )
    )
   )
  )
 ))

CONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req4:
TESTING => (setq test
't)

ACTION ACTIVATED:
(setq action
'((add-c-list
   (quote
    (PTRANS (ACTOR (nil))
            (object (nil))
            (TO (nil))
            (FROM (nil))
            (TIME (PAST))
    )
   )
  )
 ))


A NEW CONCEPT ADDED TO THE C-LIST:
  CON2 is assigned for
(setq item
'(PTRANS (ACTOR (nil)) (object (nil)) (TO (nil)) (FROM (nil)) (TIME (PAST))))
```

```
CONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req3:
TESTING => (setq test
'(precede '(pp (CLASS (HUMAN))) '(PTRANS)))

ACTION ACTIVATED:
(setq action
'((fill_slot '(ACTOR object)
             (precede '(pp (CLASS (HUMAN))) '(PTRANS))
             '(PTRANS)
  )
 ))


CONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req2:
TESTING => (setq test
'(follow '(pp (CLASS (LOCATION))) '(PTRANS)))

CONSIDERING THE ACTIVE REQUESTS...
 No more request


READ THE NEXT WORD: HOME
LOAD THE REQUESTS FROM THE DICTIONARY
The entry for HOME is:
(setq entry
'(((TEST: t) (ACTION: ((add-c-list '(pp (CLASS (LOCATION)) (NAME (HOME)))))))))


req5 IS REQUEST IDENTIFER FOR:
(setq req-id
'((TEST: t) (ACTION: ((add-c-list '(pp (CLASS (LOCATION)) (NAME (HOME))))))))

CONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req5:
TESTING => (setq test
't)

ACTION ACTIVATED:
(setq action
'((add-c-list '(pp (CLASS (LOCATION)) (NAME (HOME))))))


A NEW CONCEPT ADDED TO THE C-LIST:
  CON3 is assigned for
(setq item
'(pp (CLASS (LOCATION)) (NAME (HOME))))


OONSIDERING THE ACTIVE REQUESTS...
CONSIDERING req2:
TESTING => (setq test
'(follow '(pp (CLASS (LOCATION))) '(PTRANS)))

ACTION ACTIVATED:
(setq action
'((fill_slot '(TO) (follow '(pp (CLASS (LOCATION))) '(PTRANS)) '(PTRANS))))


CONSIDERING THE ACTIVE REQUESTS...
 No more request

***ALL WORDS PROCESSED***
FINAL RESULT:
(setq f-res
'(PTRANS (FROM (nil))
         (TIME (PAST))
         (ACTOR (pp (CLASS (HUMAN)) (NAME (JOHN))))
         (object (pp (CLASS (HUMAN)) (NAME (JOHN))))
         (TO (pp (CLASS (LOCATION)) (NAME (HOME))))
 ))
 -> ( exit )
```

画面5-3-3 CA実行例

5-4　エキスパート・システム

知識工学の一つの成果がエキスパートシステムですが，エキスパート・シェルはそのエキスパート・システムの構築を助けるためのツールです．

ここでは，まず，エキスパート・システム構築ツールとはどういうものかをKnowledge Craft (Carnegie-Group Inc. 国内販売元：インテリジェント　テクノロジー社)を例に用いて説明します．

Knowledge Craft(以下KC)は，カーネギーメロン大学で開発された知識表現言語SRL (Schema Representation Language)の後継言語CRL(Carnegie Representation Language)を核に構成されたシステムです．ただし，動作環境はsymbolics 3600等の上であって，PC-9801ではありません．ここでは，エキスパート・システム・シェルの概念を理解するために特別にAIマシン上でのシステムを使って説明します．

5-4-1　知識表現

一般にエキスパート・システムというと，if～then～型のルール(プロダクション)を知識表現の形として構成されているプロダクション・システムがほとんどです．

しかし，KCでは知識表現言語CRL(フレーム型)，CRL-OPS(プロダクション型)，CRL-PROLOG(述語論理型)の混合型となっています．

5-4-2　スキーマ

知識を表現する方法の一つにスキーマという方法があります．

スキーマというのは，ある知識をその属性と値から構成する表現方法で，例えばTechknow 98 Vを表現すると，

スキーマ名(表現する知識の名)　：　Techknow 98 V

属性　：　値

種類	：	本
価格	：	××××円
出版社	：	BNN

図5-4-1　Techknow 98 Vのスキーマ

という形になります．

図5-4-2に示したのは，KCで使われている様な本格的スキーマの例です．属性を記述する部分をスロットといい，その値がスロット値になります．

プログラム 5-4-1 スキーマの例

```
{{ make-cpul-board-spec
        is-a : engineering-activity specification-development
        sub-activity-of : develop-board-cpul
        initial-activity-of : develop-board-cpul
        expected-completion-date : "August 8,1985"
        initiated : t
        completed : nil
        description : "Develop specification for the cpu board
}}
```

スキーマは一般に，その名とスロット，スロット値で構成されています．上の例では，

スキーマ　：make-cpu 1-board-spec
スロット　：is-a など
スロット値：enginearing-activity など

となっています．

5-4-3 インヘリタンス

人間の持つ知識は多くの場合，階層的になっています．例えば，Techknow-88 も 98 V も同じ本という種類の物です．この様なときに，各々のフレームに本の持つ属性，例えば紙で作られている等，を書くことは非常に無駄になります．この様な知識の階層性はインヘリタンスという考え方で効率的に扱うことができます．

つまり，スキーマのスロットに本であるということが書かれていたなら，Techknow-98 V の材料を調べるには，本を表現したスキーマを探せばよく，Techknow-98 V 自体のスキーマには書かなくてよいのです．このような階層的な継承をインヘリタンスと呼びます．

フレーム型の表現であるスキーマを用いるとインヘリタンスが扱い易くなります．図 5-4-2 のスキーマでは make-cpu 1-board-spec は is-a スロットによって ,enginearing-activity の一種であると定義されています．

このことにより，特定の指定がなければ，このスキーマはその上位スキーマである enginearing-activity の持つ特徴が継承されます．

5-4-4 プロダクション・ルール

KC では，フレーム型表現の他に，CRL-OPS によってプロダクション型(if～then～型)の表現を扱える様になっています．プロダクション・ルールを次に示します．

プログラム　プロダクション・ルール

```
(p enable-rule
  { <activity>
    (engineering-activity ^completed t
                          ^schema-name <comleted-activity>)}
  { <next-activity>
    (engineering-activity ^enable-by <completed-activity>
                          ^initiated nil)}
  --->
  (   modify <next-activity> ^initiated t))
```

プロダクション・ルールでは，LHS(条件部)が成立するとRHS(実行部)が実行されて，推論が進行します．簡単な例を示すと，

```
(Techknow 98 v
            (発売元='BNN)          ← LHS
            →(print '正解)         ← RHS
)
```

この場合，発売元という変数の値が 'BNNならLHSが成立し，正解と表示されます．

5-4-5　述語論理

日本の国家プロジェクトであるICOTの第5世代計画では，その基盤をprologなどに見られる様なロジック・プログラミングに置いています．

KCでは，CRL-PROLOGによって述語論理による表現を行っています．述語論理による表現を次に示します．述語論理の詳しい解説はprologについて書かれた本を参考にしてください．

プログラム 5-4-3　CRL-PROLOGによる述語論理表現

```
(late-activity <x> <t2>) <
        (<x> ^is-a activity)
        (<x> ^expected-completion-date <t1>)
        (<y> ^enable-by <x> ^initiation-date <t2>)
        (< <t2><t1>)
        (bind <t2>(print-time '<t2>)
```

5-4-6 Tiny-expert

ここでは極めて簡単なプロダクション型エキスパート・システム・シェルをサンプルとして上げておきます．仕様は次のとおりです．

・最大ルール数 ……50

・LHS ………………5

・RHS ………………1

・変数は持たせない

・LHSの成立／否定は，ユーザーがY／Nを入力して決定する

プログラム 5-4-4　Tiny-expert

```
/* expert.c                                                          */
/*                                                                   */
/*                     Expert System Experimental Set                */
/*                     update 1984. copyright (C) DMSC               */
/*                                                                   */

#include "stdio.h"
#define MAXRULE 50      /* ルールの数を最大５０に設定 */
struct rule {
      char lhs[5][40];          /* LHS */
      char rhs[40];             /* RHS */
      int ifn;
      int used;
      } rules[MAXRULE];

int rulen, factn, failn, nth;
char facts[50][40], fails[50][40];

main(argc,argv)
int argc;
char *argv[];
{
factn = failn = 0;
      readin(argv[1]);
      top();
}

readin(s)
char s[];
{
int i, ifi, then, j;
      FILE *fopen(), *fp;
      char linbuf[40];
i = j = 0;
ifi = 0;
then = 0;

      if ((fp = fopen(s,"r")) == NULL) {
            printf("Can not open file¥n");
            exit(0);
            }

      while(i < MAXRULE && (*linbuf != '*')) {
```

```
            fgets(linbuf,40,fp);
                if(strcmp(car(linbuf),"if¥n") == 0) {
                        ifi = 1;
                        then = 0;
                        }
              else if (strcmp(car(linbuf),"then¥n") == 0) {
                        ifi = 0;
                        then = 1;
                        }
              else if(strcmp(car(linbuf),"end¥n") == 0) {
                        rules[i].ifn = j;
                        then = j = ifi = 0;
                        rules[i].used = 0;
                        i++;
                        }
              else if(strcmp(car(linbuf),"*¥n") == 0) {
                        printf("Rule loading finished¥n");
                        }
              else if(ifi == 1) {
                        strcpy(rules[i].lhs[j],linbuf);
                        j++;
                        rules[i].ifn = j;
                        }
              else if(then == 1) {
                        strcpy(rules[i].rhs,linbuf);
                        }
               }
            rulen = i;
            fclose(fp);
}


top()
{
   loop:
      if(nth < rulen) {
              if(rules[nth].used == 0){
                      if (eval_lhs(nth) == 1){
                              do_rhs(nth);
                              }
                      }
                      nth++;
                      goto loop;
              }
      else{
              printf("All rule used... finishing inference¥n");
              }
}

eval_lhs(r)
int r;
{
int i, rr;
      i = 0;
      rules[r].used = 1;
  evalbody:
      if(i < rules[r].ifn){
              if (exist(rules[r].lhs[i]) == 1){
                      i++;
                      goto evalbody;
                      }
              else if (existf(rules[r].lhs[i]) == 1) {
                      return(0);
                      }
```

```
			else if ((rr = backsearch(rules[r].lhs[i])) != 0) {
				if(eval_lhs(rr) == 1) {
					addfact(rules[r].lhs[i]);
						}
				else {
					addfail(rules[r].lhs[i]);
					return(0);
					}
				}
			else {
				if (qa(rules[r].lhs[i]) == 1){
					addfact(rules[r].lhs[i]);
					i++;
					goto evalbody;
					}
				else{
					addfail(rules[r].lhs[i]);
					return(0);
					}
				}
			}
		else
			return(1);
}

do_rhs(r)
int r;
{
	printf("Rule Deduces: %s¥n",rules[r].rhs);
	addfact(rules[r].rhs);
}

qa(qrule)
char *qrule;
{
	printf("Is this true: %s ¥n",qrule);
	printf("Input Yes or No:");
	return(yesno());
}


existf(s)
char s[];
{
int i;
	for(i = 0; i < failn; i++) {
		if(strcmp(s,fails[i]) == 0)
			return(1);
		}
	return(0);
}

exist(s)
char s[];
{
int i;
	for(i = 0; i < factn; i++) {
		if(strcmp(s,facts[i]) == 0)
			return(1);
		}
	return(0);
}
```

```
addfact(s)
char s[];
{
      strcpy(facts[factn++],s);
}

addfail(s)
char s[];
{
      strcpy(fails[failn++],s);
}

yesno()
{
int c, i;
char s[10];
i = 0;
      while((c = getchar()) != '¥n')
              s[i++] = c;
      if(s[0] == 'y' || s[0]=='Y')
              return(1);
      else
              return(0);
}

backsearch(r)
char *r;
{
int i;
      for(i = 0; i < rulen; i++) {
              if((strcmp(rules[i].rhs,r) == 0)
                      && (rules[i].used == 0)){
                              return(i);
                              }
              }
      return(0);
}


car(s)
char *s;
{
char carbuffer[80],*t;
int pshift;
t = &carbuffer;
      pshift = 0;
      while(*(s+pshift) != ' ' && *(s+pshift) != '¥0' && pshift < 78) {
              *(t+pshift) = *(s+pshift);
              pshift++;
              }
              *(t+pshift) = '¥0';
              return(t);
}

cdr(s)
char *s;
{
char cdrbuffer[80], *t;
int pshift, tpshift;
t = &cdrbuffer;
      pshift = 0;
      tpshift = 0;
      while(*(s+pshift) != ' ' && *(s+pshift) != '¥0')
```

```
            pshift++;
        if (*(s+pshift) == ' ') {
            pshift++;
        while(*(s+pshift) != '¥0') {
            *(t+tpshift) = *(s+pshift);
            tpshift++;
            pshift++;
            }
        }
        *(t+tpshift) = '¥0';
        *(t+79) = '¥0';
        return(t);
}
```

各ルールは，40文字の文字列からなる条件／実行部節から構成されています．

ルールの評価は，その文字列によって示されていることが正しいか否かのみの簡単なものです．しかし，インタープリタの小型のものを作ることができる人ならば，変数を扱えるシステムへ拡張することも可能でしょう．

推論は前向きに進行します．つまり，if 部が成立すれば，then 部を実行するという形です．if 部の真偽はファクト・テーブル(facts[ ])，及びフェイル・テーブル(fails[ ])を見て決定しますが，どちらにも存在しないときには，その LHS 節を RHS に持つルールを指します．これは，LHS を目標として後向きの推論を試みていることになります．しかし，該当するルールが発見されなかったときには，ユーザーに質問して真偽を確認します．

5-4-6-1 実行方法

Tiny-expert は非常に簡単なシステムであり，通常の MS-DOS 環境でそのまま使用することができます．

起動するには次のように入力します．

```
A>EX <ルールファイル名>
```

例えば，

```
A>EX CAR.RUL
```

これは〝CAR.RUL〞というファイルを使って推論します．

起動されると，Tiny-expert は推論を開始します．入力が必要なときには，

```
Is this true :  <ルール条件>
Yes or No :
```

と聞いてくるので，Y または N を入力してください．すべての可能な推論が終了した時点で，MS-DOS に制御を戻します．

車名	ドアの数	ルーフの有無	排気量	ターボ	DOHC
クラウン	4ドア	あり	3000cc	×	○
スープラ	2ドア	なし	3000cc	○	○
ソアラ	2ドア	あり	3000cc	○	○
マークII	4ドア	あり	2000cc	○	○
コロナ	4ドア	あり	2000cc	×	○
カリーナ	2ドア	あり	1800cc以下	○	○
カローラ	4ドア	あり	1800cc以下	×	×
スターレット	2ドア	あり	1800cc以下	○	○

注意：表中の車名は実在の車名とは全く関係ありません

上表をエキスパートシステムのルールファイルとした例を以下に示します。

```
if
4 door car
A roof is needed
3000litter car
No turbo
DOHC
then
crown
end
if
2 door car
No roof car
3000litter car
Turbo
DOHC
then
supra
end
if
2 door car
A roof is needed
3000litter car
Turbo
DOHC
then
soara
end
if
4 door car
A roof is needed
2000litter car
Turbo
DOHC
then
mark-II
end
if
4 door car
A roof is needed
2000litter car
No turbo
DOHC
then
colona
end
if
2 door car
A roof is needed
Less than 1800l
Turbo
DOHC
then
carina
end
if
4 door car
A roof is needed
Less then 1800l
No turbo
No DOHC
then
carora
end
if
2 door car
A roof is needed
Less than 1800l
Turbo
DOHC
then
starlet
end
*
```

図 5-4-2 ルールの実例

```
B:¥>expert car.rul
Rule loading finished
Is this true: 4 door car

Input Yes or No:y
Is this true: A roof is needed

Input Yes or No:y
Is this true: 3000litter car

Input Yes or No:y
Is this true: No turbo

Input Yes or No:y
Is this true: DOHC

Input Yes or No:y
Rule Deduces: crown

Is this true: 2 door car

Input Yes or No:N
Is this true: 2000litter car

Input Yes or No:N
Is this true: Less then 1800l

Input Yes or No:N
All rule used... finishing inference
```

画面5-4-1 プログラムの実行例

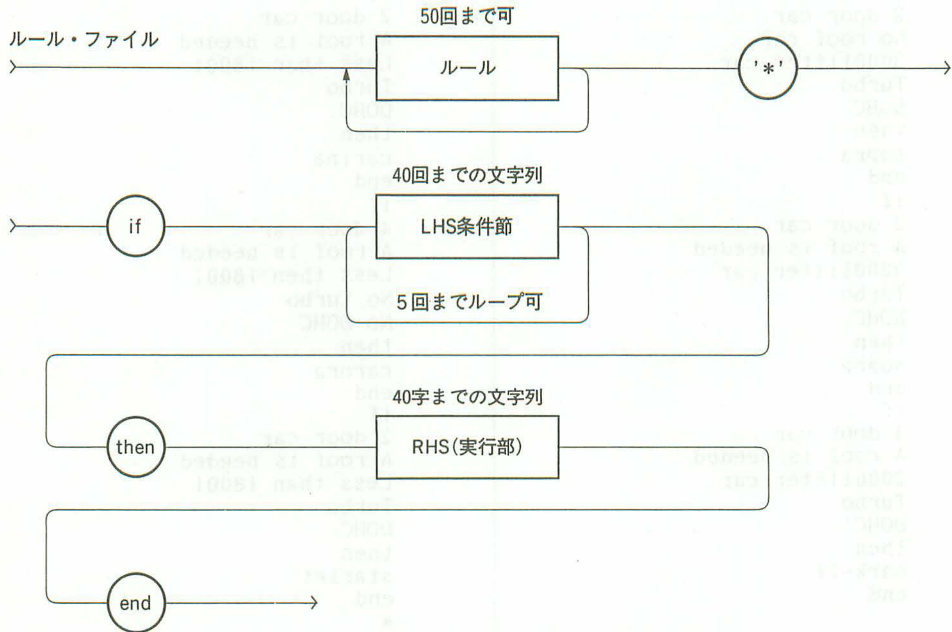


図5-4-3 ルールの書き方

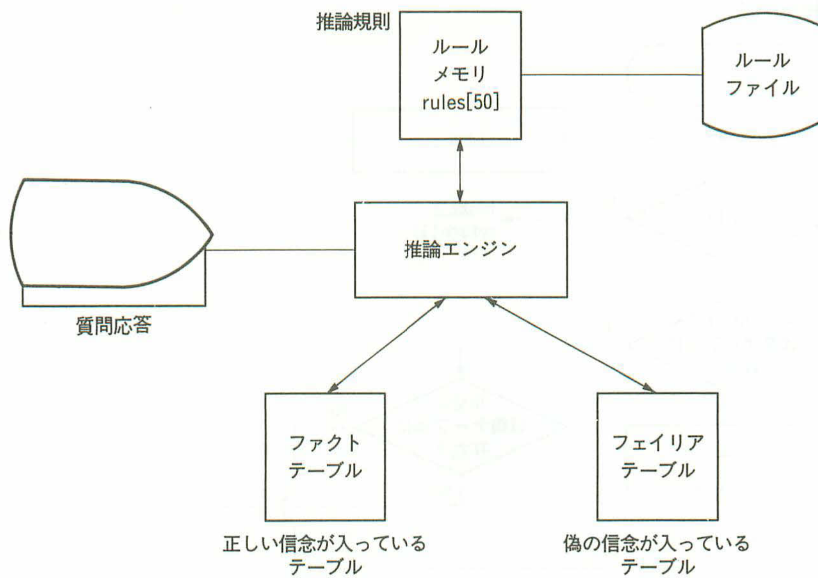
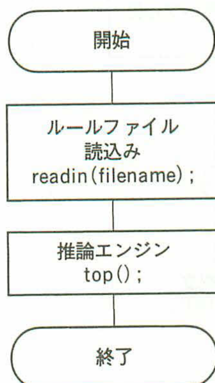
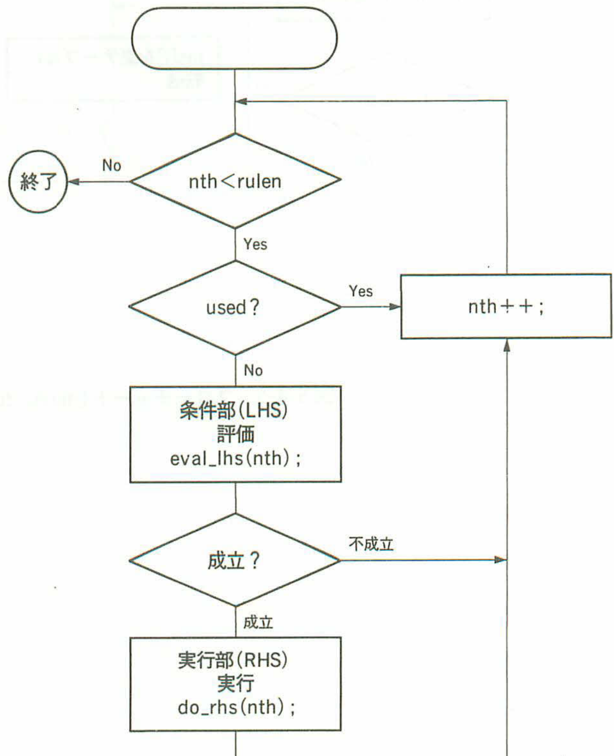


図 5-4-4　システム構成

■main



■top



■eval_lns

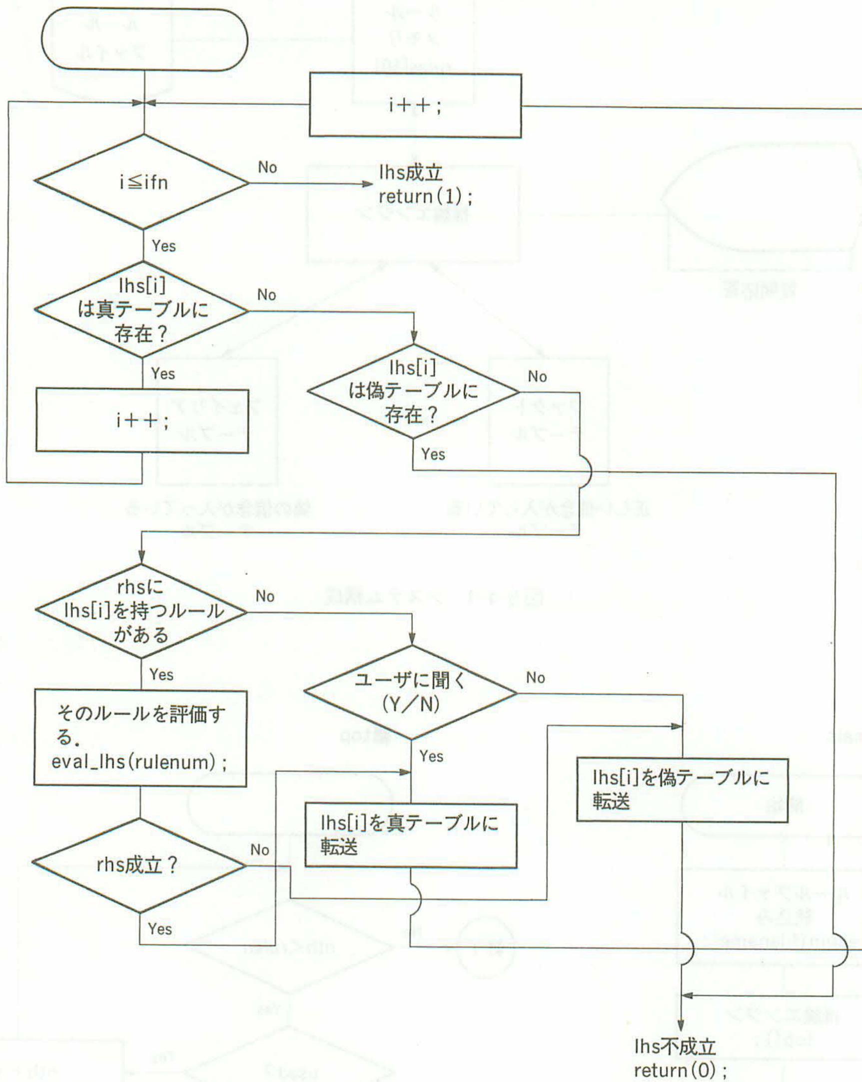


図5-4-5　フローチャート(main, top, eval-lhs)

TK-SHELL version 1.0

第6章

6-1 TK-SHELL の解説

6-1-1 コマンド・シェルについて

MS-DOS のプロンプトから実行できるコマンドは，外部コマンドと内部コマンドの 2 種類に分けることができます．

外部コマンドは MASM や C 言語を使って増やすことができ，DOS 上のプログラムの大半はこの外部コマンドとして扱うことができます．

内部コマンドは，COMMAND.COM というコマンド・インタープリタによって処理が行われます．内部コマンドの変更を行う方法は，このコマンド・インタープリタを作り変えるか，シェルと呼ばれるコマンド処理プログラムを使ってコマンドの受け渡しを行う 2 つが一般的な手法です．CPU やハードウェアと OS，コマンド・インタープリタ，シェルが，どのような関係を持っているかを図 6-1-1 に示します．

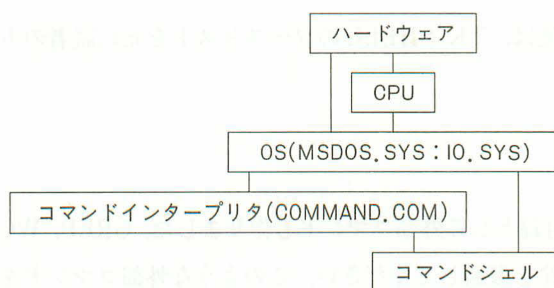


図 6-1-1　コマンドシェルの位置付け

この章では，コマンドシェルを設計しプログラミングする手法を解説していきます．DOS 上でのプログラミング環境，オペレーション環境を自由に変更し，より使いやすくしていくことは，何ら特別な MS-DOS の利用法ではありません．むしろ，個々の環境や使用状況に OS を合わせていくほうが，パーソナルコンピュータ利用法として自然でしょう．

6-1-2 TK-SHELL の仕様

MS-DOS のコマンド・インタプリタ(COMMAND. COM)は，UNIX を意識して設計されており，それだけでも十分な機能をもっています．ただ，不満な点として HISTORY 機能(以前に実

行されたコマンドを再実行する機能)とALIAS機能(頻繁に使うコマンドを，利用者が使いやすいような名前に再定義する機能)があげられます．

また，TYPEコマンドで表示するファイルは，複数のファイル名をコマンドラインに列記できると便利です．DISK-BASICを使用した経験のある利用者ならば，マルチステートメント機能(複数のコマンドを列記し，順番に実行していく機能)も便利に感じるでしょう．

＜付加された機能＞

・HISTORY機能

・ALIAS機能

・操作するファイル名の列記

・マルチステートメント機能

紙面の都合上，COMMAND．COMの便利な機能が削除されてしまいました．

＜削除された機能＞

・パイプ機能

・リダイレクトによるアペンド機能

・バッチコマンド内での構造

・プロンプトの設定

・日付・時間を表示する機能

これらの削除された機能は，TK-SHELLのソースリストを元に読者の方々がご自分で補って頂けると幸いです．

6-1-3　外部コマンド

また，TK-SHELLの付録として外部コマンドも作りました．GREP，WCがそれですが，詳しい機能や使用法は，第2節を参照してください．このような外部コマンドをいくつか加えることによって，TK-SHELLの環境はより使いやすく便利になります．

6-2 TK-SHELLの使用法

6-2-1　起動法

HI-TECH Cコンパイラによって生成されたTKSH．EXEファイルがカレントドライブのカレントディレクトリにあることを確認し，MS-DOSのプロンプトから画面6-2-1のように入力しリターンキーを押してください．

プログラムをディスクから読み込み，しばらくするとメッセージを表示しプロンプトが変更されます．これで，TK-SHELLの様々な機能を利用することができます．

この際に起動しているCOMMAND.COMのバージョンは2.11もしくは3.10のいずれかである必要があります.

```
A>TKSH

Tk-SHELL ver 1.0        copyright (C) 1986 DMSC.

A:TKSH>
```

画面 6-2-1 TK-SHELL の起動

6-2-2 内部コマンドリファレンス

TK-SHELL 内部で処理されるコマンドの使用法を解説します.

ALIAS(エリアス)

機能

・頻繁に使用するコマンド名を利用者によって再定義します.

・再定義されたコマンド名のリストを表示します.

入力 A:TKSH>ALIAS [定義前のコマンド名] [コマンドに定義する名前]

パラメータを省略した場合,再定義されたコマンド名のリストを表示します.

解説

頻繁に使用するコマンド名を短い名前に再定義することによって,タイプ量を減らすことができます.もしくは,分かりやすい名前で再定義し,初心者のOS利用を簡単にするという効果も期待できます.再定義できるコマンドは最大20個までで,名前のなかに/(スラッシュ),;(セミコロン),TAB(タブコード),スペースを含むことはできません.また,再定義するコマンド名は,最長64文字までです.

BREAK(ブレーク)

機 能

・ブレーク(CTRL+C)コードのチェック機能の有無を設定します.

・ブレーク(CTRL+C)コードのチェック機能の状態を表示します.

入 力　A：TKSH>BREAK [ON／OFF]

パラメータを省略した場合，ブレーク(CTRL+C)コードのチェック機能の状態を表示します.

解 説

初期状態において，ブレークコードのチェックはプロンプトが表示されている場合のみ行われるように設定されています.このことは，ディスクアクセスやアセンブルなどの処理中は，CTRL+Cによる処理の中断ができないようになっているということです.BREAKコマンドによってBREAKをONの状態に設定すると，すべてのMS-DOSの機能(ファンクション・コール)を利用する度に行われるようになります.このコマンドは，MS-DOS内部コマンドのBREAKと全く同一の機能を持っています.

CAT(キャット)

機 能

ファイルの内容を表示します.

入 力　A：TKSH>CAT [ファイル名1] [ファイル名2]・・・[ファイル名n]

解 説

CATに続く，すべてのファイルをディスプレイに出力します.TYPEコマンドと同じ機能をもっています.

CHDIR(チェンジ・ディレクトリ)

機 能

・カレント・ディレクトリを別のディレクトリに変更します.

・その時のカレント・ディレクトリのディレクトリ名を表示します.

入　力　A：TKSH>CHDIR　[パス名]
A：TKSH>CD　[パス名]

パラメータを省略した場合，その時のカレント・ディレクトリのディレクトリ名を表示します．

解　説
階層ディレクトリによってディスクファイルの構造が作られている場合，現在注目しているディレクトリのことをカレント・ディレクトリと呼びます．ファイル名にパス名が付けられなかった場合，このカレント・ディレクトリ内で処理が実行されます．
CHDIRコマンドは，このカレント・ディレクトリを変更する命令です．

CLS(クリア・スクリーン)

機　能
画面上に表示されているテキストデータを消去します．

入　力　A：TKSH>CLS

解　説
MS-DOSのCLSコマンドと同一の機能です．

COPY(コピー)

機　能
ファイルのコピーを作成します．

入　力　A：TKSH>COPY　[ファイル名1]　[ファイル名2]・・・[ファイル名n(ディレクトリ名)]

解　説
最終パラメータによって渡されたファイル名(ディレクトリ)に，それ以前に書かれたファイルをすべてコピーします．その際，ワイルドカードの使用はできません．具体的な使用法は，実例を見てください．

実 例

A>COPY FILE 1 :FILE 1をカレントディレクトリにコピー

A>COPY FILE 1 FILE 2 :FILE 1をFILE 2にコピー

A>COPY FILE 1 DIR 1 :FILE 1をDIR 1¥FILE 1にコピー

A>COPY FILE 1 FILE 2 FILE 3 DIR 1:FILE 1, FILE 2, FILE 3をDIR 1にコピー

DEL(デリート)

機 能

ファイルを削除します.

入 力 A:TKSH>DEL [ファイル名1] [ファイル名2]・・・[ファイル名n]

解 説

DELに続く,すべてのファイルを消去します.ワイルドカードの使用はできません.ERASEコマンドと同じ機能をもっています.

DIR(ディレクトリ)

機 能

ディレクトリの内容を表示します.

入 力 A:TKSH>DIR [ディレクトリ名] [/W][/A]

解 説

ディレクトリの内容を表示します.ディレクトリ名を列記しても末尾に書かれたディレクトリの内容しか表示しません.スイッチは[/W]と[/A]の2つがあり,[/W]は80桁モードでファイル名のみ表示し,[/A]は1行に1つずつファイル名とファイルサイズを表示します.何もスイッチを付けなかった場合,1行に1つずつファイル名が表示されます.ファイル属性のチェックをしていないので,通常は見ることのできないMS-DOS.SYSやIO.SYSなどのファイルも表示されます.ワイルドカードを使用することはできません.

ERASE(イレース)

機　能

ファイルを消去します.

入　力　A：TKSH>ERASE ［ファイル名1］［ファイル名2］・・・［ファイル名n］

解　説

ERASEに続く，すべてのファイルを消去します．ワイルドカードの使用はできません．DELコマンドと同じ機能をもっています.

EXIT(イグジット)

機　能

チャイルドプロセスとして起動されたCOMMAND.COMから，親プロセスに戻ります.

入　力　A：TKSH>EXIT

解　説

TK-SHELLからチャイルドプロセスとしてCOMMAND.COMを呼び出した場合，TK-SHELLに戻るためのコマンドとして使用します．TK-SHELLが動作している状態でEXITコマンドを実行すると，TK-SHELLから抜けてCOMMAND.COMに戻ります.

HELP(ヘルプ)

機　能

TK-SHELLの内部コマンドの簡単な説明が表示されます.

入　力　A：TKSH>HELP

解　説

HELPコマンドによって，コマンドリファレンスが表示されます.

HISTORY(ヒストリ)

機　能

・それ以前に実行されたコマンドのリストを表示します.

・それ以前に実行されたコマンドをコマンドリストの番号で指定し，そのコマンドを実行します.

入　力　A：TKSH＞HISTORY　[コマンド番号]

解　説

それ以前に実行したコマンドのリストを表示することによって，うっかり忘れてしまったコマンド名やタイプミスを防ぐことができます．また，何度も実行する処理はALIASコマンド同様，タイプ量を減らすためにHISTORYを使うことによって簡単に行うことができます．HISTORYに登録されるコマンドは最大10個までで，リターンキーを押す度に内容は更新されます.

MKDIR(メイク・ディレクトリ)

機　能

新しいディレクトリを作成します.

入　力　A：TKSH＞MKDIR　[新しく作成するディレクトリ名]

A：TKSH＞MD　[新しく作成するディレクトリ名]

解　説

このコマンドは階層ディレクトリの構造を作成するために使用します.

PWD(プリント・ワークディレクトリ)

機　能

その時のカレント・ディレクトリのディレクトリ名を表示します.

入　力　A：TKSH＞PWD

解　説

パラメータを持たないCHDIRと同じ機能を持っています.

RENAME(リネーム)

機　能

ファイル名を変更します.

入　力　A：TKSH>RENAME　[ファイル名1]　[ファイル名2]

A：TKSH>REN　[ファイル名1]　[ファイル名2]

解　説

指定されたファイルのファイル名1をファイル名2に変更します．ファイル名1とファイル名2のパスは一致していなければなりません.

RMDIR(リムーブ・ディレクトリ)

機　能

階層ディレクトリを削除します.

入　力　A：TKSH>RMDIR　[ディレクトリ名]

A：TKSH>RD　[ディレクトリ名]

解　説

MKDIRコマンドで作成された階層ディレクトリを削除します．ファイルが存在するディレクトリやルートディレクトリは削除することができません.

SET(セット)

機　能

・環境文字変数の値を設定します.

・現在設定されている環境文字変数を表示します.

入　力　A：TKSH>SET　[文字変数名]=[文字列]

パラメータが省略された場合，現在設定されている環境文字変数を表示します.

解　説

アプリケーションプログラムなどで使用する環境変数は，プログラムセグメントプレフィクスに渡されています．この機能を使ってプログラム間で情報のやりとりを行うことができます．また，

COMMAND.COMの内部コマンドであるPATHコマンドがTK-SHELLでは用意されていないので，このSETコマンドを使って設定してください.

実 例

A：TKSH>SET PATH =¥BIN；　　　：パスの設定

A：TKSH>SET TEMP　　　：TEMPという環境変数をクリア

TYPE(タイプ)

機 能

ファイルの内容を表示します.

入 力　A：TKSH>TYPE ［ファイル名1］［ファイル名2］・・・［ファイル名n］

解 説

TYPEに続く，すべてのファイルをディスプレイに出力します.CATコマンドと同じ機能を持っています.

VERIFY(ベリファイ)

機 能

・ディスクへの書き込みの際に，正しく書き込まれたかどうか検査を行うスイッチのON／OFFを設定します.

・現在のベリファイ・スイッチの状態を表示します.

入 力　A：TKSH>VERIFY ［ON／OFF］

パラメータが省略された場合，現在のベリファイ・スイッチの状態を表示します.

解 説

ベリファイ・スイッチをONにしておくと，ディスクにデータを書き込むごとにその内容の検査を行います.

VER(バージョン)

機 能

その時に使用されているCOMMAND. COMのバージョンを表示します.

入 力　A：TKSH＞VER

解 説

TK-SHELLは，MS-DOS version 2.11,3.10上での動作が保証されています．

6-2-3 COMMAND．COMの起動

TK-SHELL上で，COMMAND．COMをチャイルドプロセスとして起動する方法は，プロンプトからCOMMANDと打ち込むだけです．TK-SHELLに戻る場合はEXITで戻ってください．

6-2-4 マルチステートメント機能

複数のコマンドを一度の入力で行う場合，TK-SHELLではマルチステートメント機能が使用できます．コマンドは〝；〞によって区切って入力してください．左に書かれたコマンドから順番に実行していきます．次に例を示します．

A：TKSH＞DIR A： ／A ； DIR B： ／A ； DIR C： ／A

6-2-5 バッチファイルの扱い

バッチファイルは，プロンプトから入力するコマンドをファイルから読み込んで実行するものです．TK-SHELLでは，バッチファイルの中にラベルによる構造(GOTO文など)を使うことはできません．

6-2-6 外部コマンドリファレンス

外部コマンドのGREP，WCの使用法を説明します．

GREP(グレップ)

機 能

ファイル内の文字列を検索します．

入 力　A：TKSH＞GREP ［検索文字列］［ファイル名］［-VCLNHY］

解 説

COMMAND．COM上で実行した場合，ファイル名にワイルドカードが使用できますが，TK-SHELL上ではワイルドカードを使用することはできません．また，いずれの場合においても，検索文字列にワイルトカードを使用することはできません．

<オプションの説明>

-V ：マッチしなかった行をすべて表示します

-C ：ファイル内でマッチした行数を表示します

-L ：マッチした行を含むファイル名を表示します

-N ：マッチした行と行番号を表示します

-H ：ファイル名のヘッダを表示しません

-Y ：大文字小文字の区別なく検索をします

WC(ワードカウンタ)

機　能

ファイル内の文字列，文字数，行数を表示します．

入　力　A：TKSH>WC ［ファイル名］［-WCL］

解　説

COMMAND. COM 上で実行した場合，ファイル名にワイルドカードが使用できますが，TK-SHELL 上ではワイルドカードを使用することはできません．

<オプションの説明>

-W ：スペースで区切られた文字列の数を数えます

-C ：ファイルの行数を数えます

-L ：ファイルの文字数を数えます

6-3 TK-SHELL プログラムソースリスト

6-3-1 コンパイルの仕方

TK-SHELL，WC，GREP とも，すべて HI-TECH C によってコンパイルします．

● TK-SHELL の場合

ディスクに TKSH.C，MSUB.AS があることを確認してから，

```
A>C TKSH.C MSUB.AS
```

としてください．

● GREP, WCの場合

ディスクにGREP.C，またはWC.Cがあることを確認してから，

A>C GREP.C (または, WC.C) -O -R

としてください．

6-3-2　ソースリスト

TKSH.Cのソースリストの中には，多くの注釈を挿入しましたので，それをみて解析してみてください．

プログラム 6-3-1　GREP.C

```
/* grep.c
                    text serch program for HI-TECH C compiler
                     Programmed by BOGY. copyright (C) DMSC
                     grep [-options..] text [[file.name]..]
                                c find.c -r -o

          options..
                -v ﾏｯﾁ ｼﾅｲ ｷﾞｮｳ ｦ ｽﾍﾞﾃ ﾋｮｳｼﾞ ｽﾙ｡
                -c ﾌｧｲﾙ ﾁｭｳ ﾃﾞ ﾏｯﾁ ﾗｲﾝｽｳ ｦ ﾋｮｳｼﾞ ｽﾙ｡
                -l ﾏｯﾁ ｽﾙ ﾗｲﾝ ｦ ﾌｸﾑ ﾌｧｲﾙ ﾈｰﾑ ｦ ﾋｮｳｼﾞ ｽﾙ｡
                -n ﾏｯﾁ ｽﾙ ﾗｲﾝﾄ  ﾗｲﾝ ﾅﾝﾊﾞ ｦ ﾋｮｳｼﾞ ｽﾙ｡
                -h ﾌｧｲﾙ ﾈｰﾑ ﾉ ﾍｯﾀﾞ ｦ ﾋｮｳｼﾞ ｼﾅｲ｡
                -y ｵｵﾓｼﾞ ｺﾓｼﾞ ﾉ ｸﾍﾞﾂﾅｸ ﾏｯﾁ ｻｾﾙ｡

*/

#include        "stdio.h"
#include        "ctype.h"

#define         MAX_TEXT80
#define         MAX_ARGS300

int     v=0, c=0, l=0, n=0, h=1,y=0;

main(argc,argv)
int     argc;
char    *argv[];
{
        FILE    *fp,*fopen();
        int     i,j,k,line,count;
        char    *text,*new_argv[MAX_ARGS];
        char    get_text[1000];

        extern char     **_getargs();
        extern int      _argc_;

        if (argc==1) {
                argv=_getargs(0,"?");
                argc=_argc_;
        }

        text=0;         /* init */

        for (j=0,i=1; i<argc; i++) {
```

```
		if (argv[i][0]=='-') {
			k=1;
			while(argv[i][k]!=0) {
				switch(upper(argv[i][k++])) {
					case 'V': { v=1; break; }
					case 'C': { c=1; break; }
					case 'L': { l=1; break; }
					case 'N': { n=1; break; }
					case 'H': { h=0; break; }
					case 'Y': { y=1; break; }
				}
			}
		}
		else if (text==0) text=argv[i];
		else {
			new_argv[j]=argv[i];
			j++;
		}
	}
	if (text==0) {
		printf("?¥n");
		exit(1);
	}

	for (i=0 ; i<j; i++) {
		if ((fp=fopen(new_argv[i],"r"))==NULL) continue;
		count=0;
		line=0;
		while(!(feof(fp) || ferror(fp))) {
			fgets(get_text,1000,fp);
			line++;
			if (myinstr(get_text,text)) {
				count++;
				print(new_argv[i],line,get_text);
			}
			else printe(new_argv[i],line,get_text);
		}
		fclose(fp);
		printc(new_argv[i],count);
	}
}

myinstr(s,f)
char  *s,*f;
{
	char	*ff;
	ff=f;
	while(!(*f==0 || *s==0)) {
		if (y) {
			if (upper(*s)==upper(*f)) { s++; f++; }
			else if (f==ff) s++;
			else f=ff;
		}
		else {
			if (*s==*f) { s++; f++; }
			else if (f==ff) s++;
			else f=ff;
		}
	}
	if (*f!=0) return(0);
	return(1);
}
```

```
print(fn,line,text)
int   line;
char  *fn,*text;
{
      if (c || l || v) return;
      if (h) printf("%s:",fn);
      if (n) printf("%d:",line);
      printf("%s",text);
      return;
}

printe(fn,line,text)
int   line;
char  *fn,*text;
{
      if ( v == 0 ) return;
      if (h) printf("%s:",fn);
      if (n) printf("%d:",line);
      printf("%s",text);
      return;
}

printc(fn,count)
char  *fn;
int   count;
{
      if (l) { if (c) printf("%s:",fn);
              else  printf("%s",fn);
      }
      if (c) printf("%d",count);
      if (c || l) printf("¥n");
      return;
}

upper(a)
char  a;
{
      if ('a'<=a && a<='z') a-=('a'-'A');
      return(a);
}
```

プログラム6-3-2 WC.C

```
/*
            word counter for MS-DOS for HI-TECH C compiler.
               Programmed by BOGY. copyright (C) DMSC
                       cc wc.c -o -r
*/

#include "stdio.h"
#define       NULL     0

main(argc,argv)
int   argc;
char  *argv[];
{
      int     line=0,chr=0,word=0,fc=0;
      int     add_line=0,add_chr=0,add_word=0;
      int     i,j,w=1,l=1,c=1;
      FILE    *fp;
      char    work[2000];
```

```
        extern char     **_getargs();
        extern int      _argc_;

        if (argc==1) {
                argv=_getargs(0,"?");
                argc=_argc_;
        }

        for ( i=1; i<argc; i++) {
                if (argv[i][0]=='-') {
                        j=1;
                        w=0;
                        l=0;
                        c=0;
                        while (argv[i][j]!=0) {
                                switch(upper(argv[i][j++])) {
                                        case 'W': { w=1; break; }
                                        case 'L': { l=1; break; }
                                        case 'C': { c=1; break; }
                                }
                        }
                }
                else {
                        if ((fp=fopen(argv[i],"r"))==NULL) continue;
                        while(!(feof(fp) || ferror(fp))) {
                                fgets(work,2000,fp);
                                line++;
                                chr +=char_count(work);
                                word+=word_count(work);
                        }
                        fclose(fp);
                        fc++;
                        printlwc(line,word,chr,argv[i],l,w,c);
                        add_line+=line;         line=0;
                        add_chr+= chr;          chr=0;
                        add_word+=word;         word=0;
                }
        }
        if ( fc != 1 )
                printlwc(add_line,add_word,add_chr,"total",l,w,c);
}
char_count(s)
char    *s;
{
        int     a=0;
        while(*(s++)!='¥0') a++;
        return(a+1);
}

word_count(s)
char    *s;
{
        int     a=0,b=0;
        while(*s!='¥0') {
                if (!(*s==' ' || *s=='¥t' || *s==',' || *s=='¥n' ||
                        *s=='.' || *s=='=')) { b=1; s++; continue; }
                if (b==1) { a++; b=0; }
                s++;
                /* str. separated by tab,sapce,comma,equql,period ... */
        }
        if (b==1) a++;
        return(a);
}
```

```
printlwc(line,word,chr,fn,l,w,c)
int   line,word,chr,l,w,c;
char  *fn;
{
      if (l) printf("%5d ",line);
      if (w) printf("%5d ",word);
      if (c) printf("%5d ",chr);
      printf("%s¥n",fn);
}

upper(a)
char  a;
{
      if ('a'<=a && a<='z') a-=('a'-'A');
      return(a);
}
```

プログラム 6-3-3 TKSH.C

```
/* tksh.c                                              */
/* Tk-shell bsh program for MS-DOS 2.11 , 3.10        */
/*                           Programmed by BOGY */
/*     Update 1986 09 30    copyright (C) DMSC         */

#include "stdio.h"
#include "ctype.h"
#include "signal.h"
#include "setjmp.h"

#defineupper(g) (('a'<=g && g<='z')?g-('a'-'A'):g)
#defineERROR    -1
#defineHIS_MAX  10                          /* history max */
#defineALIAS_MAX 20                     /* alias max */

#defineSTDIN    0
#defineSTDOUT   1
#defineSTDERR   2

static int      env_max = 0x1000;          /* ｶﾝｷｮｳ ﾉ ｾｯﾃｲ ﾁ */
static int      env_f     = 1;
static int      slac      = 0;
static char     *env;

static int      func_d  = 1;                    /* ﾌｸﾞﾗﾑ ﾊ ｽｸﾞ ｼｭｳﾘｮｳ */
static int      exit_f    = 1;
static char     old_buff[128];
static char     inp_buff[128];
static char     his_buff[ HIS_MAX ][80];
static char     ali_buff[ ALIAS_MAX*2][64];

static int      bat=0;                          /* bat file exec flag */
static FILE     *batin;
static int      sv_stdin=0xffff;  /* save stdin,out with redirect */
static int      sv_stdout=0xffff;
static int      sv_stderr=0xffff;

extern char     ** environ ;

#defineMAXARGS           30
#defineARGSLINE 20
```

```
static char     _argtmp[MAXARGS*ARGSLINE+MAXARGS*2];
static int      uargc;
static jmp_buf  jb_err;

struct ioregsw {
                int     ax;
                int     bx;
                int     cx;
                int     dx;
                int     si;
                int     di;
        };

struct ioregsb {
                unsigned char   al,ah;
                unsigned char   bl,bh;
                unsigned char   cl,ch;
                unsigned char   dl,dh;
        }
union   regs {
                struct  ioregsb h;
                struct  ioregsw x;
        };

union   regs    inregs;
union   regs    outregs;

int     cbreak(),cchdir();
int     ccopy()  ,cdel() ,cdir();
int     cexit()  ,cmkdir(),cmkdir();
int     crename(),crmdir(),cset();
int     ctype(),cverify();
int     cvol(),chistory(),calias();
int     intdos(),chelp();
int     cver(),ccls(),cecho();

static struct   {
                        char    *name;
                        int     (*func)();
                } com[]= {
                        "BREAK"         ,cbreak,
                        "CHDIR"         ,cchdir, "CD"   ,cchdir,
                        "PWD"           ,cchdir,
                        "CLS"           ,ccls,
                        "COPY"          ,ccopy,
                        "DEL"           ,cdel,   "ERASE"        ,cdel,
                        "DIR"           ,cdir,
                        "ECHO"          ,cecho,
                        "EXIT"          ,cexit,
                        "MKDIR"         ,cmkdir, "MD"   ,cmkdir,
                        "RENAME" ,crename, "REN"   ,crename,
                        "RMDIR"         ,crmdir, "RD"   ,crmdir,
                        "SET"           ,cset,
                        "TYPE"          ,ctype,  "CAT"          ,ctype,
                        "VERIFY" ,cverify,
                        "VER"           ,cver,
                        "VOL"           ,cvol,
                        "HISTORY" ,chistory,
                        "ALIAS"         ,calias,
                        "HELP"          ,chelp,
                        "*"             ,cexit
                };
```

```
main(argc,argv)
char    *argv[];
int     argc;
{
        void       intr();
        int        i,c;
        char       his_temp[128];

        signal(SIGINT,intr);
        interr();

        for (i=1; i<argc; i++) init(argv[i]);
        if (func_d != 1 ) exit(0);
        if (env_f) init_env(env_max);
        printf("¥nTk-shell ver 1.0¥tcopyright (C) 1986 DMSC.¥n¥n");

        for ( i=0; i<HIS_MAX; i++) his_buff[i][0]=0;
        for ( i=0; i<ALIAS_MAX; i++)
                ali_buff[i*2][0]=ali_buff[i*2+1][0]=0;

        while(exit_f) {
                setjmp(jb_err);
                c=msdos(0x1900)&0xff;
                printf("%c:tksh>",'A'+c);
                if (bat) {
                        if ( ferror(batin) || feof(batin)
                           || fgets(inp_buff,128,batin)==0 ) {
                                fclose(batin);
                                bat=0;
                                printf("¥n");
                                continue;
                        }
                        dellf(inp_buff);
                        printf("[bat]:%s¥n",inp_buff);
                }
                else {
                        if (gets(inp_buff)==0) {
                                cexit();
                                continue;
                        }
                }
                strcpy(his_temp,inp_buff);
                command();
                shistory(his_temp);
        }
        cexit();
}

intr()
{
        unredirect();
        longjmp(jb_err,0);
}

dellf(s)
char    *s;
{
        while( *s != 0) {
                if (*s=='¥n') *s=' ';
                s++;
        }
        return;
}
```

```
init(s)
char    *s;
{
        printf("%s",s);
        if ( *s != '/' ) {
                printf("paramater error. %s¥n",s);
                exit(1);
        }
        switch( upper(*(s+1)) ){
                case 'C':
                        if (env_f) {
                                init_env(env_max);
                                env_f=0;
                        }
                        command(s+2);
                        cexit(0);
                case 'E':
                        if (*(s+2)!=':') {
                                printf("paramater error. %s¥n",s);
                                exit(1);
                        }
                        sscanf(s+3,"%4x",&env_max);
                        if (env_max<0x200 || env_max>0x7fff) env_max=0x1000;
                        break;
                case 'P':
                        slac=1;
                        break;
        }
        return;
}

init_env(a)
int     a;
{
        char    **senv;
        char    *tenv,*tmp;
        char    *malloc();

        env=(char *)(( (int)(tmp=malloc(a)) +15)&0xfff0);
        if (tmp==0) {
                printf("memory allocation error.¥n");
                exit(1);
        }
        tenv=env;
        for ( senv = environ; *senv; senv++) {
                tmp=*senv;
                while(*(tenv++)=*(tmp++)) ;
        }
        *tenv=0;
        return;
}

char    *
cgetenv(s)
char    *s;
{
        register char *   xp;
        char    *zskip();
        int     i;
        i = strlen(s);
        for(xp = env ; *xp ; xp = zskip(xp))
                if ( ustrncmp(xp, s, i) == 0 && xp[i] == '=')
                        return ( xp + i + 1 );
```

```
        return( 0 );
}

ustrncmp(s,d,n)
char    *s,*d;
int     n;
{
        int         k;
        for (k=0; k<n; k++) {
                if ( upper(*s) != upper(*d) ) return(1);
                else {
                        s++; d++;
                }
        }
        return(0);
}

upline(s)
char    *s;
{
        while(*s = upper(*s)) s++;
        return;
}

char    *
zskip(s)            /* skip until zero */
char    *s;
{
        while(*(s++)!=0);
        return(s);
}

char    *
zback(s)            /* skip back until zero */
char    *s;
{
        while((*s != 0) && (s != env) ) s--;
        if (s == env) {
                return(env);
        }
        else {
                return(s+1);
        }
}

cdelenv(s)
char    *s;
{
        char      *t,*d,*cgetenv(),*zskip(),*zback();
        if ( (t=cgetenv(s)) == 0 ) return(-1);
        d=zback(t);
        t=zskip(t);
        zcopy(d,t);
        return(0);
}

zcopy(d,s)
char    *d,*s;
{
        if (*s==0) {
                *d=0;
                return;
        }
```

```
        while((*(d++)=*(s++))+*s != 0 );
        *d=0;
        return;
}

cprtenv()
{
        char    *s,*zskip();
        for ( s = env; *s != 0; s=zskip(s)) {
                while(*s!='=') putchar(*(s++));
                printf("¥t= %s¥n",s+1);
        }
        return;
}

csetenv(s1,s2)
char    *s1,*s2;
{
        char    *t,*zskip();
        cdelenv(s1);
        t=env;
        while(*t!=0) t=zskip(t);
        sprintf(t,"%s=%s%c%c",s1,s2,0,0);
        return;
}

command()
{
        /* commmand ｼﾞｯｺｳ */
        int     i,r;
        int     exec();
        char    *a,*index();
        char    **b;
        char    **ugetargs();
        char    *skipsepa(),*ustrcmp(),*index(),*ustrcpy();
        char    path[64],name[64],nm[64],*j;

        do {
                a=inp_buff-1;
                while(1){
                        a=index(a+1,';');
                        if ( a == 0 ) break;
                        if ( *(a+1) == ';' ) {
                                strcpy(a,a+1);
                                a++;
                        }
                        else break;
                }

                if ( a != 0 ) {
                        if (*(a+1)==';') {
                                strcpy(old_buff,skipsepa(a+2));
                                *(a+1)=0;
                        }
                        else {
                                strcpy(old_buff,skipsepa(a+1));
                                *a=0;
                        }
                }
                else    old_buff[0]=0;

                ealias(inp_buff);
```

```
                    /* change drive */
if (inp_buff[1]==':' && inp_buff[2]==0) {
        cchdrive();
        inp_buff[0]=0;
}

/* command exec */

redirect(inp_buff);

for( i=0; com[i].name[0]!='*'; i++) {
        if ((a=ustrcmp(com[i].name,inp_buff)) !=0 ) {
                b=ugetargs(a,"?");
                (*(com[i].func))(uargc,b);
                inp_buff[0]=0;
                break;
        }
}

if (inp_buff[0]!=0) {
        j=cgetenv("PATH");
        if ( j == 0 ) path[0]=0;
        else strcpy(path,j);
        b=ugetargs(inp_buff,"?");
        strcpy(nm,b[1]);
        if (index(nm,'¥¥')==0) {
        while( (r=exec(uargc,b))!=0 && path[0]!=0 ) {
                if ((j=index(path,';'))==0) {
                        strcpy(name,path);
                        if (name[strlen(name)-1]!='¥¥') {
                                strcat(name,"¥¥");
                        }
                        strcat(name,nm);
                        path[0]=0;
                }
                else {
                        j=ustrcpy(name,path);
                        if (name[strlen(name)-1]!='¥¥') {
                                strcat(name,"¥¥");
                        }
                        strcat(name,nm);
                        if (*j==0) path[0]=0;
                        else strcpy(path,j+1);
                }
                b=ugetargs(inp_buff,"(com line)");
                strcpy(nm,b[1]);
                b[1]=name;
        }
        if (r!=0) {
                printf("?:file not found.¥n");
        }
        }
        else {
                if (exec(uargc,b)!=0) {
                        perror("?:file not found.¥n");
                }
        }
        inp_buff[0]=0;
}
unredirect();
strcpy(inp_buff,old_buff);
printf("¥n");
```

```
	} while(inp_buff[0]!=0 );
	return;
}

redirect(s)
char	*s;
{
	char	*a,*b,*c,*d,in_file[64],out_file[64];
	int	app=0,err=0,in,out;

	char	*skiptosepa(),*skipsepa(),*index();

	in=0;
	out=0;

	if ( index(s,'>')==0 && index(s,'<')==0 ) {
		return;
	}
	if ( (a=index(s,'<')) != ( char * ) 0 ) {
							/* save stdin */
		inregs.h.ah=0x45;
		inregs.x.bx=STDIN;
		if ( intdos(&inregs,&outregs) ) {
			printf("Can't make file-handle.¥n");
			intr();
		}
		sv_stdin=outregs.x.ax;
		b=skipsepa(a+1);
		if (*b==0) {
			printf("bad redirect.");
			intr();
		}
		c=skiptosepa(b);
		if (*c==0) {
			*a=0;
			strcpy(in_file,b);
		}
		else {
			*c=0;
			strcpy(in_file,b);
			strcpy(a,c+1);
		}
						/* open in_file */
		inregs.x.ax=0x3d00;
		inregs.x.dx=( int )in_file;
		if (intdos(&inregs,&outregs)==1) {
			perror("?");
			intr();
		}
		in=outregs.x.ax;
						/* file_handle copy */
		inregs.h.ah=0x46;
		inregs.x.bx=in;
		inregs.x.cx=STDIN;
		if (intdos(&inregs,&outregs)==1) {
			perror("?");
			intr();
		}
						/* close */
		inregs.x.bx=in;
		inregs.h.ah=0x3e;
```

```
                if (intdos(&inregs,&outregs)==1) {
                        perror("?");
                        intr();
                }
        } else sv_stdin=0xffff;

        if ( (a=index(s,'>')) != ( char * ) 0 ) {
                                                        /* save stdout */
                inregs.h.ah=0x45;
                inregs.x.bx=STDOUT;
                if ( intdos(&inregs,&outregs) ) {
                        printf("Can't make file-handle.¥n");
                        intr();
                }
                sv_stdout=outregs.x.ax;

                d=a+1;
                switch(*d){
                        case '>':
                                app=1;
                                d++;
                        case '&':
                                err=1;
                                d++;
                }
                if (err ==1 ) {
                        inregs.h.ah=0x45;
                        inregs.x.bx=STDERR;
                        if ( intdos(&inregs,&outregs) ) {
                                printf("Can't make file-handle.¥n");
                                intr();
                        }
                        sv_stderr=outregs.x.ax;
                }


                b=skipsepa(d);
                if (*b==0) {
                        printf("bad redirect.");
                        intr();
                }
                c=skiptosepa(b);
                if (*c==0) {
                        *a=0;
                        strncpy(out_file,b,63);
                }
                else {
                        *c=0;
                        strncpy(out_file,b,63);
                        strncpy(a,c+1,127);
                }

                                                /* file make */
                if (!exist(out_file)) {
                        inregs.h.ah=0x3c;
                        inregs.x.dx=(int) out_file;
                        inregs.x.cx=0;
                        if (intdos(&inregs,&outregs)==1) {
                                perror("?");
                                intr();
                        }
                }
                                                /* open out_file */
```

```
		inregs.x.ax=0x3d02;
		inregs.x.dx=( int )out_file;
		if (intdos(&inregs,&outregs)==1) {
			perror("?");
			intr();
		}
		out=outregs.x.ax;
						/* append ? */
		if (app==1) {
			inregs.x.ax=0x4202;
			inregs.x.cx=0;
			inregs.x.dx=0;
			inregs.x.bx=out;
			if (intdos(&inregs,&outregs)==1) {
				perror("?");
				intr();
			}
		}
						/* file_handle copy */
		inregs.h.ah=0x46;
		inregs.x.bx=out;
		inregs.x.cx=STDOUT;
		if (intdos(&inregs,&outregs)==1) {
			perror("?");
			intr();
		}

		if (err==1) {
			inregs.x.cx=STDERR;
			if (intdos(&inregs,&outregs)==1) {
				perror("?");
				intr();
			}
		}
						/* close */
		inregs.x.bx=out;
		inregs.h.ah=0x3e;
		if (intdos(&inregs,&outregs)==1) {
			perror("?");
			intr();
		}
	}
	else sv_stdout=0xffff;
	return;
}

unredirect()
{
	sv_stdin  = closeredi(sv_stdin,STDIN);
	sv_stdout = closeredi(sv_stdout,STDOUT);
	sv_stderr = closeredi(sv_stderr,STDERR);
	return;
}

closeredi(std1,std2)
{
	char	tmp[2];

	tmp[0]=0x1a;
	tmp[1]=0;

	if ( std1 != 0xffff ) {
						/* close std_in */
		write(std2,tmp,1);
```

```
                inregs.h.ah=0x46;
                inregs.x.bx=std1;
                inregs.x.cx=std2;
                intdos(&inregs,&outregs);

                inregs.h.ah=0x3e;
                inregs.x.bx=std1;
                intdos(&inregs,&outregs);

        }
        return ( 0xffff );
}

char    *
ustrcpy(d,s)
char    *s,*d;
{
        while( *s!=0 && *s!=';' ) *(d++)=*(s++);
        *d=0;
        return(s);
}

char    *
ustrcmp(s,d)
char    *s,*d;
{
        while( upper(*s) == upper(*d) && *s!=0) { s++; d++; };
        if (*s != 0) return( 0 );
        if (*d == 0 || *d == ' ' || *d == '/' || *d == '¥t' ) return( d );
        return( 0 );
}

cbreak(argc,argv)
int     argc;
char    *argv[];
{
        if (argc==1) {                  /* break ﾉ ｼﾞｮｳﾀｲ ﾋｮｳｼﾞ */
                inregs.x.ax = 0x3300;
                intdos(&inregs,&outregs);
                if ( outregs.h.al == 0xff) {
                        printf("intdos exec error¥n");
                        return;
                }
                printf("break = ");
                if (outregs.h.dl == 1){
                        printf("on¥n");
                }
                else                            printf("off¥n");
                return;
        }
        if (argc==2) {                  /* break ｼﾞｮｳﾀｲ ﾉ ｾｯﾃｲ */
                if (ustrcmp("ON",argv[1])!=0) {
                        inregs.x.ax=0x3301;
                        inregs.x.dx=1;
                        intdos(&inregs,&outregs);
                        if ( outregs.h.al != 0xff) return;
                        else {
                                printf("paramater set error¥n");
                                return;
                        }
                }
                else if (ustrcmp("OFF",argv[1])!=0) {
                        inregs.x.ax=0x3301;
                        inregs.x.dx=0;
```

```
                        intdos(&inregs,&outregs);
                        if ( outregs.h.al != 0xff) return;
                        else {
                                printf("paramater set error¥n");
                                return;
                        }
                }
                else    printf("paramater error¥n");
        }
        return;
}

cchdir(argc,argv)
int     argc;
char    *argv[];
{
        char    w[64];
        if (argc==1) {
                inregs.h.ah=0x47;
                inregs.x.si=(int )w;
                inregs.h.dl=0;
                intdos(&inregs,&outregs);
                if (w[0]==0) {
                        printf("¥¥n");
                }
                else {
                        printf("%s¥n",w);
                }
                return;
        }
        else if (argc==2) {
                if (chdir(argv[1])!=ERROR) return ;
                else {
                        printf("exec error¥n");
                        return;
                }
        }
        else printf("paramater error¥n");
        return;
}

ccls()




{
        printf("%c[2J",0x1b);
        return;
}

ccopy(argc,argv)
int     argc;
char    *argv[];
{
        int     i,t,s,d,b;
        char    w[64],strend(),*rindex(),*indx,tmp[BUFSIZ];

        if (argc==1) return;
        if (argc==2) {
                argc=3;
                argv[2]=".";
```

```
        }
        for (i=1; i<argc-1; i++) {
                w[0]=0;
                if ( isdir(argv[argc-1]) ) {
                        if ( strend(argv[argc-1]) == '¥¥' )
                                strcpy(w,argv[argc-1]);
                        else {
                                strcpy(w,argv[argc-1]);
                                strcat(w,"¥¥");
                        }
                        if ( (indx=rindex(argv[i],'¥¥')) != (char *) 0 ||
                                                        ( indx = rindex(argv[i],'

                                strcat(w,indx+1);
                        } else {
                                strcat(w,argv[i]);
                        }
                }
                else {
                        strcpy(w,argv[argc-1]);
                }

                if ((s=open(argv[i],0))==ERROR) {
                        perror("?");
                        continue;
                }

                if ((d=creat(w))==ERROR) {
                        perror("?");
                        continue;
                }

                while(1) {
                        t=read(s,tmp,BUFSIZ);
                        if (t==0) break;
                        b=write(d,tmp,t);
                        if (t!=b) {
                                perror("?");
                                break;
                        }
                }
                close(s);
                close(d);
        }
        return;
}
char    *
bsearch(s,c)
char    *s,c;
{
        char      *p;
        p=s+strlen(s);
        while( p!=s && *p!=c) p--;
        return(p);
}

char
strend(s)
char    *s;
{
        char      *p;
        p=s;
        while(*p!=0) p++;
        if ( p == s ) return(0);
```

```
        return( *(p - 1) );
}

isdir(s)
char    *s;
{
        char      dta[128];
        if (index(s,'*')!=0 || index(s,'?')!=0) return(0);
        if (*s=='¥¥' && *(s+1)==0) return(1);
        if (*(s+1)==':' && *(s+2)=='¥¥' && *(s+3)==0 ) return(1);
        msdos(0x1a00,dta); /* set dta */
        inregs.h.ah=0x4e;
        inregs.x.dx=( int ) s;
        inregs.x.cx=0x10;
        if (intdos(&inregs,&outregs)==1) return(0);
        return(1);
}

cdel(argc,argv)
int     argc;
char    *argv[];
{
        int       i;

        for (i=1; i<argc; i++)
                if (unlink(argv[i])==ERROR) {
                        printf("%s can't delete.¥n",argv[i]);
                        perror("?");
                }
        return;
}

cdir(argc,argv)
int     argc;
char    *argv[];
{
        int       a,wi,d,i,s;
        unsigned char       dta[128],w[70],tmp[30];
        unsigned long int   f;

        a=wi=d=0;
        for (i=1; i<argc; i++) {
                if (argv[i][0]=='/') {
                        switch( upper(argv[i][1]) ) {
                                case 'W':
                                        wi=i;
                                        break;
                                case 'A':
                                        a=i;
                                        break;
                        }
                }
                else {
                        d=i;
                }
        }
        if ( d == 0 ) {
                strcpy(w,"*.*");
        }
        else {
                if (isdir(argv[d])==1) {
                        if (strend(argv[d])=='¥¥') {
                                strcpy(w,argv[d]);
                                strcat(w,"*.*");
```

```
                        }
                        else {
                                strcpy(w,argv[d]);
                                strcat(w,"¥¥*.*");
                        }
                }
                else if (argv[d][1]==':' && argv[d][2]==0) {
                        strcpy(w,argv[d]);
                        strcat(w,"*.*");
                }
                else    {
                        strcpy(w,argv[d]);
                }
        }
        msdos(0x1a00,dta); /* set dta */
        inregs.h.ah=0x4e;
        inregs.x.dx=( int ) w;
        inregs.x.cx=0x17;
        if (intdos(&inregs,&outregs)==1) return;
        inregs.h.ah=0x4f;
        do {
                printf("%s",dta+0x1e);
                for (i=0; i<(16-strlen(dta+0x1e)); i++) putchar(' ');
                if (wi==0 && a!=0) {
                        s=1;
                        tmp[0]=0;
                        if ((dta[0x15] &  0x1)==0x1 ) strcat(tmp,"[RO]");
                        if ((dta[0x15] &  0x2)==0x2 ) strcat(tmp,"[HID]");
                        if ((dta[0x15] &  0x4)==0x4 ) strcat(tmp,"[SYS]");
                        if ((dta[0x15] &  0x8)==0x8 ) {
                                s=0;
                                strcat(tmp,"[VOL]");
                        }
                        if ((dta[0x15] & 0x10)==0x10) {
                                s=0;
                                strcat(tmp,"[DIR]");
                        }
                        if ( s != 0 ) {
                                for (f=0,i=0x1d; i>=0x1a; i--) {
                                        f=f*(long)256 + (long)dta[i];
                                }
                                printf("%10D",(long)f);
                        }
                        else {
                                printf("          ");
                        }
                        printf("  %s",tmp);
                }
                if ( wi == 0 ) printf("¥n");
                inregs.x.dx=( int ) w;
                inregs.x.cx=0x17;
                inregs.h.ah=0x4f;
        } while((msdos(0x1a00,dta),intdos(&inregs,&outregs))==0 ) ;
        return;
}

cexit()
{
        free(env);
        if ( slac == 0 ) exit_f=0;
        return;
}
```

```
cmkdir(argc,argv)
int     argc;
char    *argv[];
{
        if (argc!=2) return;
        if (mkdir(argv[1])==ERROR){
                printf("Can't make (%s) dir.¥n",argv[1]);
                return;
        }
        return;
}

crename(argc,argv)
int     argc;
char    *argv[];
{
        if (argc!=3) return;
        if (rename(argv[1],argv[2])==ERROR) {
                perror("?");
        }
        return;
}

crmdir(argc,argv)
int     argc;
char    *argv[];
{
        if (argc!=2) return;
        if (rmdir(argv[1])==ERROR) perror("?");
        return;
}

cset(argc,argv)
int     argc;
char    *argv[];
{
        if (argc==1) {
                cprtenv();
                return;
        }
        if (argc==3) {
                csetenv(argv[1],argv[2]);
                return;
        }
        if (argc!=2) return;
        cdelenv(argv[1]);
        return;
}

ctype(argc,argv)
int     argc;
char    *argv[];
{
        int     a;
        FILE    *fp;

        for (a=1; a<argc; a++) {
                if ((fp=fopen(argv[a],"r"))==(struct _iobuf *)ERROR) continue;
                while(!(feof(fp)||ferror(fp))) putchar(fgetc(fp));
                fclose(fp);
        }
        return;
}
```

```
cverify(argc,argv)
int     argc;
char    *argv[];
{
        if (argc==1) {                  /* verify ﾉ ｼﾞｮｳﾀｲ ﾋｮｳｼﾞ */
                inregs.x.ax = 0x5400;
                intdos(&inregs,&outregs);
                printf("verify = ");
                if (outregs.h.al == 1){
                        printf("on¥n");
                }
                else    printf("off¥n");
                return;
        }
        if (argc==2) {                  /* verify ｼﾞｮｳﾀｲ ﾉ ｾｯﾃｲ */
                if (ustrcmp("ON",argv[1])!=0) {
                        inregs.x.ax=0x2e01;
                        intdos(&inregs,&outregs);
                }
                else if (ustrcmp("OFF",argv[1])!=0) {
                        inregs.x.ax=0x2e00;
                        intdos(&inregs,&outregs);
                }
                else    printf("paramater error¥n");
        }
        return;
}

cver()
{
        int     a;
        a=msdos(0x3000);
        printf("MS-DOS Ver %d.%d¥n",a&0xff,a/256 );
        return;
}

cvol()
{
        printf("VOL¥n");
        return;
}

chistory(argc,argv)
int     argc;
char    *argv[];
{
        int     i,t,n;
        if (argc==1) {      /* display history stack */
                for (i=1; i<HIS_MAX; i++) {
                        printf("%d¥t%s¥n",i,his_buff[i]);
                }
                return;
        }
        else if (argc==2) {
                sscanf(argv[1],"%d",&i);
                if (i<0 || i>HIS_MAX ) return;
                strcpy(old_buff,his_buff[i-1]);
        }
        else if (argc==3) {
                for (n=0,t=0,i=1; i<argc; i++) {
                        if (argv[i][0]=='-' && upper(argv[i][1])=='T') {
                                t=i;
                        }
                        else n=i;
```

```
                }
                if (n==0 || t==0) {
                        printf("?¥n");
                        return;
                }
                sscanf(argv[n],"%d",&i);
                if (i<0 || i>HIS_MAX) return;
                strcpy(inp_buff,his_buff[i-1]);
                intr();
        }
        return;
}

shistory(s)
char    *s;
{
        int     i;
        for (i=0; i<HIS_MAX; i++) {
                strcpy(his_buff[i],his_buff[i+1]);
        }
        strcpy(his_buff[ HIS_MAX-1 ],s);
        return;
}

calias(argc,argv)
int     argc;
char    *argv[];
{
        int     i;

        if (argc==1) {                  /* display alias cond. */
                for (i=0; i<ALIAS_MAX; i++)
                        if (ali_buff[i*2][0]!=0)
                         printf("%s¥t%s¥n",ali_buff[i*2],ali_buff[i*2+1]);
        }
        else if (argc==2) {/* delete alias */
                for (i=0; i<ALIAS_MAX; i++) {
                        if (strcmp(ali_buff[i*2],argv[1])==0)
                                ali_buff[i*2][0]=0;
                }
        }
        else if (argc==3) {/* set alias */
                for (i=0; ali_buff[i*2][0]!=0 && i!=ALIAS_MAX; i++);
                if (i==ALIAS_MAX) {
                        printf("buffer over¥n");
                        return;
                }
                strcpy( ali_buff[i*2] , argv[1] );
                strcpy( ali_buff[i*2+1] , argv[2] );
        }
        return;
}

ealias(s)
char    *s;
{
        int     i;
        char    *d,buff[128];

        for (i=0; i<ALIAS_MAX; i++) {
                if (ali_buff[i*2][0]==0) continue;
                if ((d=ustrcmp(ali_buff[i*2+1],s))!=0) {
                        strcpy(buff,ali_buff[i*2]);
```

```
                                        strcat(buff,d);
                                        strcpy(s,buff);
                                        return;
                                }
                        }
                return;
}

cchdrive()
{
        int a;
        a=msdos(0x0e00,(upper(inp_buff[0]))-'A');
        msdos(0x0d00);                  /* reset */
        if ((upper(inp_buff[0]))-'A'<(a & 0xff)) return;
        printf("bad drive number¥n");
        return;
}

char    **
ugetargs(s,n)
char    *s,*n;
{
        char            **argv;
        int             argc,quart;
        char            *st,*d,*m,*skipsepa();
        char            buff[200],*b;

        st=_argtmp;
        d=st+MAXARGS*ARGSLINE;
        m=d+MAXARGS*2;
        argc=1;
        argv=(char **) d;
        argv[0]=n;
        quart=0;

        while(*s!=0) {
                b=buff;
                s=skipsepa(s);
                if (*s==0) break;
                while(issepa(*s)==0) *(b++)=*(s++);
                *b=0;
                strcpy(st,buff);
                argv[argc++]=st;
                st=st+strlen(st)+1;
                if (st>=m) {
                        printf("no room for string¥n");
                        uargc=argc-1;
                        return(argv);
                }
        }
        uargc=argc;
        return(argv);
}

issepa(c)
char    c;
{
        if (c==' ' || c=='¥t' || c==0 ) return(1);
        return(0);
}

char    *
skipsepa(s)
char    *s;
```

```
{
        while(issepa(*s) && *s != 0) s++;
        return(s);
}

char    *
skiptosepa(s)
char    *s;
{
        while( !issepa(*s) ) s++;
        return(s);
}

chelp()
{
        printf("      = Tk-shell HELP MENU =¥n");
        printf("BREAK      [ on/off ]¥n");
        printf("CHDIR      [ pathname ] / CD¥n");
        printf("CLS¥n");
        printf("COPY       source dest[dir]¥n");
        printf("DEL        filename ..¥n");
        printf("DIR        [ pathname , /w ,/a ]¥n");
        printf("EXIT¥n");
        printf("MKDIR      pathname / MD¥n");
        printf("RENAME     old new / REN¥n");
        printf("RMDIR      path-name¥n");
        printf("SET        [ environ value / environ ]¥n");
        printf("TYPE       filename ... ¥n");
        printf("VERIFY     [ on/off ]¥n");
        printf("VER¥n");
        printf("HISTORY    [ number ]¥n");
        printf("ALIAS      [ str replace ]¥n");
        return;
}

cecho(argc,argv)
int     argc;
char    *argv[];
{
        int     i;
        for (i=1; i<argc; i++) printf("%s¥n",argv[i]);
        return;
}

exec(argc,argv)
int     argc;
char    *argv[];
{
        char    w[0x80];
        char    c[0x80];
        int     pb[7],i,sef,k;
        char    p1[13],p2[13];

        void    uabort(),intr();
        char    *index();


        int     getds();
        int     getes();

        for (i=0; i<argc-1; i++) {
                argv[i]=argv[i+1];
        }
        argc--;
```

```
	pb[0]=getes()+( int )env / 16;
	if (argc>1) strncpy(p1,argv[1],12);
	if (argc>2) strncpy(p2,argv[2],12);
	strcpy(w,"X");
	for (i=1,k=0; i<argc; i++) {
		k=k+strlen(argv[i])+1;
		strcat(w,argv[i]);
		strcat(w," ");
	}
	strcat(w,"¥r");

	w[0]=( char )k;
	pb[1]=( int )w;
	pb[2]=getds();
	pb[3]=( int )p1;
	pb[4]=getds();
	pb[5]=( int )p2;
	pb[6]=getds();

	inregs.x.ax=0x4b00;
	strcpy(c,argv[0]);
	if ( index(c,'.') == ( char * )NULL) {
		sef=1;
		strcat(c,".COM");
	}
	else {
		sef=3;
	}
	inregs.x.dx=( int )c;
	inregs.x.bx=( int )pb;
	while ( (signal(SIGINT,uabort),intdos(&inregs,&outregs))!=0) {
		switch (sef) {
			case 1:
				strcpy(c,argv[0]);
				strcat(c,".EXE");
				sef++;
				continue;
			case 2:
				strcpy(c,argv[0]);
				strcat(c,".BAT");
				if (exist(c)) {
					execbat(c);
					sef++;
					signal(SIGINT,intr);
					return(0);
				}
			case 3:
				signal(SIGINT,intr);
				return(1);
		}
	}
	signal(SIGINT,intr);
	return(0);
}

exist(s)
char	*s;
{
	int	fd;
	if ((fd=open(s,0))==ERROR) return(0);
	close(fd);
	return(1);
}
```

```
execbat(s)
char   *s;
{
        if ((batin=fopen(s,"r"))==(FILE *)ERROR) return;
        bat=1;
        return;
}
```

プログラム6-3-4 MSUB.AS

```
;msub.as
;
;Tk-shell mashine language functions.
;Programmed by BOGY
;Update 1986 09 30copyright (C) DMSC

.title HI-TECH C: msub.as
.globlsmall_model
.psect_TEXT,class=CODE
.psectdata,class=DATA
.psectbss,class=BSS
.groupDGROUP,data,bss
.psect_TEXT

;intintdos(&inregs,&outregs);

.globl_intdos
_intdos:
pushsi
pushdi
pushbp
movbp,sp
movbx,8[bp]
movax,2[bx]
pushax
movax,[bx]
movcx,4[bx]
movdx,6[bx]
movsi,8[bx]
movdi,10[bx]
popbx
pushes
pushds
popes
int#21h
popes
pushbx
movbx,10[bp]
mov[bx],ax
mov4[bx],cx
mov6[bx],dx
mov8[bx],si
mov10[bx],di
popax
mov2[bx],ax
brcl1
movax,#0
brl2
l1:
movax,#1
```

```
l2:
movsp,bp
popbp
popdi
popsi
ret

;intgetds();

.globl_getds
_getds:
movax,ds
ret

;intgetes();

.globl_getes
_getes:
movax,es
ret

; shell ﾅｲ ﾉ error ｼｮﾘ

;voidinterr()

.globl_interr
_interr:
movdx,#_err
pushds
pushcs
popds
movax,#02524h
int#21h
popds
ret

;error trap

.globl_err
_err:
pushdx
pushsi
errs:
pushds
pushcs
popds

movdx,#errmsg
movah,#9
int#21h

movah,#1
int#21h
movcs:errax,ax

movdx,#errm2
movah,#9
int#21h
popds

movah,#0ffh
moval,cs:errax
movsi,#errno
```

```
errloop:
cmpah,cs:[si]
brzerrs
cmpal,cs:[si]
brzerrl2
incsi
incsi
brerrloop
errl2:
moval,cs:1[si]
cmpal,#2
brzerrabort
popsi
popdx
iret
errabort:
.globl_intr
jmp_intr

errno:db'A',2,'a',2
db'R',1,'r',1
db'I',0,'i',0
db0ffh

errmsg:
db'Fatal error ! R)etry , I)gnore or A)bort ? $'
errm2:
db0dh,0ah,'$'
errax:
dw0

;program abort routine

.globl_uabort
_uabort:
movax,#4c01h
int#21h
```

付録

MS-DOS(ver 3.1)システム・コール一覧

00 H　：　プログラムの終了

入力条件	リターン情報
AH ← 00 H CS　← PSP のセグメントアドレス	なし

01 H　：　キーボードから 1 文字入力

AH ← 01 H	AL ←入力文字

02 H　：　ディスプレイへ 1 文字出力

AH ← 02 H DL ←出力する文字	なし

03 H　：　補助装置から 1 文字入力

AH ← 03 H	AL ←入力文字

04 H　：　補助装置へ 1 文字出力

AH ← 04 H DL ←出力する文字	なし

05 H　：　プリンタへ 1 文字出力

AH ← 05 H DL ←出力する文字	なし

06 H　：　キーボードから 1 文字入力／ディスプレイへの 1 文字出力

AH ← 06 H DL ← 0 FFH	AL ←入力文字
AH ← 06 H DL ←出力する文字(FFH 以外)	なし

07 H ： 直接コンソール入力

AH ←07 H	AL ←入力文字

08 H ： キーボードから 1 文字入力

AH ←08 H	AL ←入力文字

09 H ： ディスプレイへの文字出力

AH ←09 H DS：DX ←文字列のオフセット	なし

0 AH ： バッファに文字列入力

AH ←0 AH DS：DX ←バッファのオフセット	なし

0 BH ： キーボード・ステータスのチェック

AH ←0 BH	AL ←00 H ：キー・バッファが空 FFH ：キー・バッファに文字が入っている

0 CH ： キーボードから 1 文字入力

AH ←0 CH AL ←01 H，06 H，07 H， 08 H，0 AH，	AL ←入力文字 00 H：AL の値が規定外

0 DH ： ディスクのリセット

AH ←0 DH	なし

0 EH ： カレント・ドライブの選択

AH ←0 EH DL ←ドライブ番号	AL ←選択されたドライブ番号

0 FH ： ファイルのオープン

AH ←0 FH DS：DX ←オープンされていない FCB	AL ←00 H ：ディレクトリエントリが存在 FFH ：ディレクトリエントリが存在しない

10 H　：　ファイルのクローズ

AH ← 10 H DS：DX ←オープンされている FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

11 H　：　最初のディレクトリエントリの検索

AH ← 11 H DS：DX ←オープンされていない FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

12 H　：　次のディレクトリエントリの検索

AH ← 12 H DS：DX ←オープンされていない FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

13 H　：　ファイルの削除

AH ← 13 H DS：DX ←オープンされていない FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

14 H　：　シーケンシャル・リード

AH ← 14 H DS：DX ←オープンされている FCB	AL ←00 H　：正常終了 01 H　：EOF 02 H　：DTA が不足している 03 H　：EOF，レコードの一部分

15 H　：　シーケンシャル・ライト

AH ← 15 H DS：DX ←オープンされている FCB	AL ←00 H　：正常終了 01 H　：ディスクに空き領域がない 02 H　：DTA が不足している

16 H　：　ファイルの作成

AH ← 16 H DS：DX ←オープンされていない FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

17 H　：　ファイル名の変更

AH ← 17 H DS：DX ←修正されたFCB	AL ←00 H　：ディレクトリエントリが存在 FFH：ディレクトリエントリが存在しない

19 H　：　カレント・ディスク番号の取得

AH ← 19 H	AL ←カレント・ディスク番号

1 AH　：　ディスク転送アドレスのセット

AH ← 1 AH DS：DX ←ディスク転送アドレス	なし

1 BH　：　デフォルト・ドライブのデータの取得

AH ← 1 BH	AL ←1クラスタ当りのセクタ数 CX ←1セクタ当りのバイト数 DX ←1ドライブ当りのクラスタ数 DS：BX ← FAT-IDのアドレス

1 CH　：　ドライブのデータの取得

AH ← 1 CH DL ←ドライブ番号	AL ← FFH：ドライブ番号が無効 FFH 以外： AL ←1クラスタ当りのセクタ数 CX ←1セクタ当りのバイト数 DX ←1ドライブ当りのクラスタ数 DS：BX ← FAT-IDのアドレス

21 H　：　ランダム・リード

AH ← 21 H DS：DX ←オープンされたFCB	AL ←00 H　：正常終了 01 H　：EOF 02 H　：DTAが不足している 03 H　：EOF，レコードの一部分

22 H　：　ランダム・ライト

AH ← 22 H DS：DX ←オープンされた FCB	AL ←00 H　：正常終了 01 H　：ディスクに空き領域がない 02 H　：DTA が不足している

23 H　：　ファイルのレコード数の取得

AH ← 23 H DS：DX ←オープンされていない FCB	AL ←00 H　：ディレクトリエントリが存在 FFH　：ディレクトリエントリが存在しない

24 H　：　相対レコードのセット

AH ← 24 H DS：DX ←オープンされた FCB	なし

25 H　：　割り込みベクトルのセット

AH ← 25 H AL ←割り込みタイプ番号 DS：DX ←割り込み処理ルーチンのオフセット	なし

26 H　：　新しい PSP の作成

AH ← 26 H DX ← PSP のセグメントアドレス	なし

27 H　：　ランダム・ブロック・リード

AH ← 27 H DS：DX ←オープンされた FCB CX ←読み出すレコード数	AL ←00 H　：正常終了 01 H　：EOF 02 H　：セグメントの終り 03 H　：EOF，レコードの一部分 CX ←読み出されたレコード数

28 H ： ランダム・ブロック・ライト

AH ←28 H DS：DX ←オープンされたFCB CX ←書き込むレコード数	AL ←00 H ：正常終了 01 H ：ディスクに空き領域がない 02 H ：セグメントの終り CX ←書き込まれたレコード数

29 H ： ファイル名の解析

AH ←29 H AL ←解析の制御(下図参照) DS：DI ←解析するストリング ES：DI ←オープンされていない FCB	AL ←00 H ：ワイルド・カードが使用されていない 01 H ：ワイルド・カード使用 FFH ：ドライブ名が無効 DS：DI ←解析されたストリングの次にくる最初のバイト ES：DI ←FCB

ビット	値	意味
0	0	ファイル分離記号を検出した場合，全ての解析を停止する
	1	先行する分離記号は無視する
1	0	ストリングにドライブ番号がない場合，FCB内のドライブ番号は0(カレント・ドライブ)にセットされる
	1	ストリングにドライブ番号がない場合，FCB内のドライブ番号は変更されない
2	0	ストリングにファイル名がない場合，FCB内のファイル名は8つのスペースにセットされる
	1	ストリングにファイル名がない場合，FCB内のファイル名は変更されない
3	0	ストリングに拡張子がない場合，FCB内の拡張子は3つのスペースにセットされる
	1	ストリングに拡張子がない場合，FCB内の拡張子は変更されない

2 AH ： 日付の取得

AH ←2 AH	CX ←年 (1980～2079) DH ←月 (1～12) DL ←日 (1～31) AL ←曜日[0(日)～6(土)]

2BH　：　日付のセット

AH ←2BH CX ←年　(1980～2079) DH ←月　(1～12) DL ←日　(1～31)	AL ←00H　：有効な日付 FFH　：無効な日付

2CH　：　時刻の取得

AH ←2CH	CH ←時(0～23) CL ←分(0～59) DH ←秒(0～59)

2DH　：　時刻のセット

AH ←2DH CH ←時(0～23) CL ←分(0～59) DH ←秒(0～59)	AL ←00H　：有効な時刻 FFH　：無効な時刻

2EH　：　ベリファイフラグのセット／リセット

AH ←2EH AL ←00H　：ベリファイ OFF 01H　：ベリファイ ON DL ←00H	なし

2FH　：　ディスク転送アドレスの取得

AH ←2FH	ES：BX ←ディスク転送アドレス

30H　：　バージョン番号の取得

AH ←30H	AL ←番号の整数部 AH ←番号の小数部 BH ←OEM のシリアル番号 BL：CX ←24 ビットのユーザー番号

31 H ： キープ・プロセス

AH ← 31 H AL ←抜け出しコード DX ←パラグラフでのメモリサイズ	なし

33 H ： <CTRL-C>チェックの取得／セット

AH ← 3 3 H AL ←00 H ：取得 01 H ：セット セットの場合 DL ←00 H ：OFF 01 H ：ON	取得の場合 DL ←00 H ：OFF 01 H ：ON AL ←FFH ：エラー(AL の値が規定外)

35 H ： 割り込みベクトルの取得

AH ← 35 H AL ←割り込み番号	ES：BX ←割り込みルーチンのオフセット

36 H ： ディスクのフリースペースの取得

AH ← 36 H DL ←ドライブ番号	BX ←使用可能なクラスタ数 DX ←1ドライブ当りのクラスタ数 CX ←1セクタ当りのバイト数 AX ←1クラスタ当りのセクタ数 FFFFH：ドライブ番号が無効

38 H ： 国別情報の取得／セット

AH ←38 H AL ←00 H ：現在の国 01 H ：USA 規格 51 H ：日本規格 DS：DX ←バッファ(32 バイト)のオフセット	キャリーフラグがセット AX ←02 H ：無効なファンクション キャリーフラグがセットされない DS：DX ←国についての情報
AH ← 38 H DX ← FFFFH AL ←カントリー・コード	キャリーフラグがセット AX ←02 H ：無効なカントリー・コード キャリーフラグがセットされない 正常終了

39 H　:　ディレクトリの作成

AH ← 39 H DS : DX ←パス名の位置	キャリーフラグがセット AX ←03 H　:無効なパス 05 H　:アクセスの否定 キャリーフラグがセットされない 正常終了

3 AH　:　ディレクトリの削除

AH ← 3 AH DS : DX ←パス名の位置	キャリーフラグがセット AX ←03 H　:無効なパス 05 H　:アクセスの否定 10 H　:現在のディレクトリ キャリーフラグがセットされない 正常終了

3 BH　:　カレントディレクトリの変更

AH ← 3 BH DS : DX ←パス名の位置	キャリーフラグがセット AX ←03 H　:無効なパス キャリーフラグがセットされない 正常終了

3 CH　:　ファイル・ハンドルの作成

AH ← 3 CH DS : DX ←パス名の位置 CX ←ファイル属性	キャリーフラグがセット AX ←03 H　:無効なパス 04 H　:オープン・ファイル過多 05 H　:アクセスの否定 キャリーフラグがセットされない AX ←ファイル・ハンドル

3DH : ファイル・ハンドルのオープン

AH ←3DH AL ←ファイル・アクセス 　　コントロール DS：DX ←パス名の位置	キャリーフラグがセット AX ←01H ：無効なファンクション 　　02H ：ファイルが存在しない 　　03H ：無効なパス 　　04H ：オープンファイル過多 　　05H ：アクセスの否定 　　0CH ：無効なアクセス キャリーフラグがセットされない AX ←ファイル・ハンドル

3EH : ファイル・ハンドルのクローズ

AH ←3EH BX ←ファイル・ハンドル	キャリーフラグがセット AX ←06H ：無効なファイル・ハンドル キャリーフラグがセットされない 正常終了

3FH : リード・ハンドル

AH ←3FH DS：DX ←バッファの位置 CX ←読み込むバイト数 BX ←ファイル・ハンドル	キャリーフラグがセット AX ←05H ：アクセスの否定 　　06H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←読み込まれたバイト数

40H : ライト・ハンドル

AH ←40H DS：DX ←バッファの位置 CX ←書き込むバイト数 BX ←ファイル・ハンドル	キャリーフラグがセット AX ←05H ：アクセスの否定 　　06H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←書き込まれたバイト数

41 H ： ディレクトリ・エントリの削除

AH ← 41 H DS：DX ←パス名の位置	キャリーフラグがセット AX ←02 H ：無効なファイル 05 H ：アクセスの否定 キャリーフラグがセットされない 正常終了

42 H ： ファイル・ポインタの移動

AH ← 42 H CX：DX ←移動するバイト数 AL ←移動方法 BX ←ファイル・ハンドル	キャリーフラグがセット AX ←01 H ：無効なファンクション 06 H ：無効な処理 キャリーフラグがセットされない DX：AX ←新規のポインタ・ロケーション

43 H ： ファイル属性の取得／設定

AH ←43 H AL ←00 H ：取得 01 H ：設定 DS：DX ←パス名の位置 CX ←ファイル属性(設定の場合)	キャリーフラグがセット AX ←01 H ：無効なファンクション 03 H ：無効なパス 05 H ：アクセスの否定 キャリーフラグがセットされない CX ←ファイル属性(取得の場合)

44 H ： デバイスに対するI／Oコントロール

AH ←44 H AL ←00 H ：データの取得 BX ←ファイル・ハンドル	キャリーフラグがセット AX ←01 H ：無効なファンクション 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない DX ←デバイス・データ
AH ←44 H AL ←01 H ：データのセット BX ←ファイル・ハンドル DX ←デバイス・データ	キャリーフラグがセット AX ←01 H ：無効なファンクション 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない DX ←デバイス・データ

AH ←44 H AL ←02 H ：データの転送 BX ←ファイル・ハンドル CX ←コントロール・データのバイト数 DS：DX ←バッファのオフセット	キャリーフラグがセット AX ←01 H ：無効なファンクション 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←転送されたバイト数
AH ←44 H AL ←03 H ：データの受け取り BX ←ファイル・ハンドル CX ←コントロール・データのバイト数 DS：DX ←バッファのオフセット	キャリーフラグがセット AX ←01 H ：無効なファンクション 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←転送されたバイト数
AH ←44 H AL ←04 H ：データの転送 BL ←ドライブ番号 CX ←コントロール・データのバイト数 DS：DX ←バッファのオフセット	キャリーフラグがセット AX ←01 H ：無効なファンクション 05 H ：無効なドライブ番号 キャリーフラグがセットされない AX ←転送されたバイト数
AH ←44 H AL ←05 H ：データの受け取り BL ←ドライブ番号 CX ←コントロール・データのバイト数 DS：DX ←バッファのオフセット	キャリーフラグがセット AX ←01 H ：無効なファンクション 05 H ：無効なドライブ番号 キャリーフラグがセットされない AX ←転送されたバイト数
AH ←44 H AL ←06 H ：入力ステータスのチェック BX ←ファイル・ハンドル	キャリーフラグがセット AX ←01 H ：無効なファンクション 05 H ：アクセスの否定 06 H ：無効なファイル・ハンドル 0 DH ：無効なデータ キャリーフラグがセットされない AX ←00 H ：非レディ状態 FFH ：レディ状態

AH ← 44 H AL ←07 H　：出力ステータスのチェック BX ←ファイル・ハンドル	キャリーフラグがセット AX ←01 H　：無効なファンクション 05 H　：アクセスの否定 06 H　：無効なファイル・ハンドル 0 DH　：無効なデータ キャリーフラグがセットされない AX ←00 H　：非レディ状態 FFH　：レディ状態
AH ←44 H AL ←08 H　：デバイスの交換性 BL ←ドライブ番号	キャリーフラグがセット AX ←01 H　：無効なファンクション 0 FH　：無効なドライブ番号 キャリーフラグがセットされない AX ←00 H　：交換可能 FFH　：交換不可能
AH ←44 H AL ←09 H　：ドライブの検出 BL ←ドライブ番	キャリーフラグがセット AX ←01 H　：無効なファンクション 0 FH　：無効なドライブ番号 キャリーフラグがセットされない DX ←デバイス・アトリビュート・ワード
AH ←44 H AL ←0AH　：ハンドルの検出 BX ←ファイル・ハンドル番号	キャリーフラグがセット AX ←01 H　：無効なファンクション 06 H　：無効なファイル・ハンドル キャリーフラグがセットされない DX ←I/O コントロール・ビットフィールド
AH ←45 H AL ←0BH　：リトライのセット BX ←リトライの回数 CX ←待ち時間	キャリーフラグがセット AX ←01 H　：無効なファンクション キャリーフラグがセットされない 正常終了

45 H ： ファイル・ハンドルの二重化

AH ← 45 H BX ←ファイル・ハンドル	キャリーフラグがセット AX ←04 H ：オープンファイル過多 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←新規のファイル・ハンドル

46 H ： ファイル・ハンドルの強制二重化

AH ← 46 H BX ←既在のファイル・ハンドル CX ←新規のファイル・ハンドル	キャリーフラグがセット AX ←04 H ：オープンファイル過多 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない AX ←新規のファイル・ハンドル

47 H ： カレント・ディレクトリの取得

AH ← 47 H DS：SI ←バッファ(64 バイト)のオフセット DL ：ドライブ番号	キャリーフラグがセット AX ←0 FH ：無効なドライブ番号 06 H ：無効なファイル・ハンドル キャリーフラグがセットされない 正常終了

48 H ： メモリの割り当て

AH ← 48 H BX ←割り当てるメモリの大きさ(パラグラフ)	キャリーフラグがセット AX ←07 H ：データの破壊 08 H ：メモリが不足 BX ←最大のメモリサイズ キャリーフラグがセットされない AX ←メモリのセグメント・アドレス

49 H ： 割り当てメモリの解放

AH ← 49 H ES ←解放するメモリのセグメントアドレス	キャリーフラグがセット AX ←07 H ：データの破壊 09 H ：無効なブロック キャリーフラグがセットされない 正常終了

4AH : 割り当てメモリ・ブロックの変更

AH ←4AH ES ←メモリのセグメントアドレス BX ←変更するメモリの大きさ (パラグラフ)	キャリーフラグがセット AX ←07H :データの破壊 08H :メモリが不足 09H :無効なブロック キャリーフラグがセットされない 正常終了

4BH : プログラムのロード/実行

AH ←4BH AL ←00H DS:DX ←パス名の位置 ES:BX ←パラメータ・ブロックの位置	キャリーフラグがセット AX ←01H :無効なファンクション 02H :ファイルが存在しない 08H :メモリが不足 0AH :不正な環境 0BH :不正なフォーマット キャリーフラグがセットされない 正常終了
AH ←4BH AL ←03H DS:DX ←パス名の位置 ES:BX ←パラメータ・ブロックの位置	キャリーフラグがセット AX ←01H :無効なファンクション 02H :ファイルが存在しない 08H :メモリが不足 0AH :不正な環境 キャリーフラグがセットされない 正常終了

4CH : プロセスの終了

AH ←4CH AL ←リターン・コード	なし

4DH : チャイルド・プロセスからのリターン・コードの受取

AH ←4DH	AX ←抜け出しコード

4 EH　：　ファイル名の検索(First)

AH ← 4 EH DS　：DX ←パス名の位置 CX ←ファイル属性	キャリーフラグがセット AX ←02 H　：ファイルが存在しない 12 H　：これ以上ファイルがない キャリーフラグがセットされない 正常終了

4 FH　：　ファイル名の検索(Next)

AH ← 4 FH	キャリーフラグがセット AX ←12 H　：これ以上ファイルがない キャリーフラグがセットされない 正常終了

54 H　：　ベリファイ状態を得る

AH ← 54 H	AL ←ベリファイフラグの値

56 H　：　ディレクトリ・エントリの変更

AH ← 56 H DS：DX ←パス名の位置 ES：DI　←新規のパス名の位置	キャリーフラグがセット AX ←02 H　：ファイルが存在しない 05 H　：アクセスの否定 11 H　：ドライブの不一致 キャリーフラグがセットされない 正常終了

57 H　：　ファイルの日付，時刻の取得／セット

AH ←57 H AL ←00 H　：取得 01 H　：セット CX ←時刻(セットの場合) DX ←日付(セットの場合)	キャリーフラグがセット AX ←01 H　：無効なファンクション 05 H　：無効なファイル・ハンドル キャリーフラグがセットされない CX ←時刻(取得の場合) DX ←日付(取得の場合)

58 H ： アロケーション・ストラテジの取得／セット

AH ←58 H AL ←00 H ：取得 AL ←01 H ：セット (セットの場合) BX ←00 H ：下位 01 H ：最小 02 H ：上位	キャリーフラグがセット AX ←01 H ：無効なファンクション キャリーフラグがセットされない (取得の場合) AX ←00 H ：下位 01 H ：最小 02 H ：上位

59 H ： エラー・コードの取得

AH ← 59 H BX ←エラー・レベル (通常 00 H)	AX ←拡張されたエラー・コード BH ←エラー・クラス BL ←可能な対処 CH ←ローカス

5 AH ： 一時ファイルの作成

AH ← 5 AH CX ←アトリビュート DS : DX ←パス名の位置	キャリーフラグがセット AX ←03 H ：パス名が存在しない 05 H ：アクセスの否定 キャリーフラグがセットされない AX ←ファイル・ハンドル

5 BH ： 新規ファイルの作成

AH ← 5 BH CX ←アトリビュート DS : DX ←パス名の位置	キャリーフラグがセット AX ←03 H ：パス名が存在しない 04 H ：オープンファイル過多 05 H ：アクセスの否定 50 H ：ファイルが既に存在する キャリーフラグがセットされない AX ←ファイル・ハンドル

5CH ： ファイル・アクセスのロック／解除

AH ←5CH AL ←00H ：ロック BX ←ファイル・ハンドル CX：DX ←ロック領域のアドレス SI ：DI ←ロック領域の長さ	キャリーフラグがセット AX ←01H ：無効なファンクション 06H ：無効なハンドル 21H ：ロックの破壊 キャリーフラグがセットされない 正常終了
AH ←5CH AL ←01H ：解除 BX ←ファイル・ハンドル CX：DX ←解除領域のアドレス SI ：DI ←解除領域の長さ	キャリーフラグがセット AX ←01H ：無効なファンクション 06H ：無効なハンドル 21H ：ロックの破壊 キャリーフラグがセットされない 正常終了

5EH ： マシン名の取得／プリンタのセットアップ

AH ←5EH AL ←00H ：マシン名の取得 DS：DX ←バッファ(16バイト)のオフセット	キャリーフラグがセット AX ←01H ：無効なファンクション キャリーフラグがセットされない CX ←ローカル・コンピュータ番号
AH ←5EH AL ←02H ：Printer Setup BX ←インデックスの割り当て CX ←セットアップ文字列の長さ	キャリーフラグがセット AX ←01H ：無効なファンクション キャリーフラグがセットされない 正常終了

5 FH　：　割り当てエントリ操作

AH ←5 FH AL ←02 H　：エントリの取得 BX ←インデックスの割り当て DS：SI　←ローカル名の位置 ES：DI　←リモート名の位置	キャリーフラグがセット AX ←01 H　：無効なファンクション 12 H　：これ以上ファイルがない キャリーフラグがセットされない BL ←03 H　：プリンタ 04 H　：ドライブ CX ←ユーザー変数域
AH ←5 FH AL ←03 H　：エントリの作成 BL ←03 H　：プリンタ 04 H　：ドライブ CX ←ユーザー変数域 DS：SI　←ソース・デバイス名の位置 ES：DI　←デスティネーション・デバイス名の位置	キャリーフラグがセット AX ←01 H　：無効なファンクション 03 H　：パス名が存在しない 05 H　：アクセスの否定 08 H　：メモリが不足 キャリーフラグがセットされない 正常終了
AH ←5 FH AL ←04 H　：エントリのキャンセル DS：DI　←ソース・デバイス名の位置	キャリーフラグがセット AX ←01 H　：無効なファンクション 0 FH　：リディレクトの中止 キャリーフラグがセットされない 正常終了

62 H　：　PSP の取得

AH ← 62 H	BX ← PSP のセグメント・アドレス

FORM64.C

```
/* form64.c
                                64byte text format program for eliza.txt
                                       copyright (C) DMSC 1986.11.01
*/

#include "stdio.h"

main(argc,argv)
int argc;
char *argv[];
{
int i , j ;
char c ;

         FILE *infile, *outfile, *fopen();

i = 0 ;

         if ( argc < 3 ) {
                    printf("Usage: FORM64 <input file> <output file>¥n");
                    return(0);
                    }

         infile = fopen(argv[1],"r");
         outfile = fopen(argv[2],"w");

         while((c = getc(infile)) != EOF ) {
                   putc(c,outfile);
                   i++;
                   if ( c == '¥n' ) {
                            for ( j = i ; j < 63 ; j++) putc('*',outfile);
                            i = 0;
                            }
                   }

         fclose(infile);
         fclose(outfile);
}
```

クイック・リファレンス

●割り込みベクタ

●1MB FDD

●640KB FDD

●1MB／640KB 両用FDD

●ハードディスク

●グラフィックBIOS

●グラフLIO

●サウンド機能

●TECH-WINDOW関数

●TK-SHELLコマンド

メディアサービスのお知らせ

本書に記載されているプログラムをディスクにて供給いたします．なお，動作に必要な環境(ハード・ソフト)はプログラムにより異なります．お買い求めになる前に確認してくださるようお願いいたします．

●対応機種

PC-9801 シリーズ

●メディア

5インチ 2DD／2HD

●価格

9,800円

●申し込み方法

巻末にとじこんである「郵便振替用紙」の空欄に，

PC-Techknow 98 V　ディスク版(メディアタイプ)

と明記し，代金 9,800 円を添えてお近くの郵便局にお申し込みください．

●注意

ディスクの発送は，**昭和62年1　月下旬**を予定しております．また，プログラムの環境動作に注意してください．

ご注意

(1)本書は著作権上の保護を受けています．本書の一部あるいは全部について，株式会社ビー・エヌ・エヌから文書による許諾を得ずに，いかなる方法においても無断で複写，複製することは禁じられています．

(2)本書についての電話によるお問い合わせはご遠慮ください．質問等がございましたら往復はがき又は切手・返信用封筒を同封の上，弊社までお送りくださるようお願いいたします．

PC-Techknow 98 V　定価 3900 円

著　者　安井　勉
執　筆　DMSC(楠見哲男・柴田幸男・他)
装　丁　木村和喜(ナチュラ)

昭和 61 年 11 月 25 日　初版発行

発行人　樺島正博
発行所　株式会社ビー・エヌ・エヌ
東京都千代田区麹町 4-5　紀尾井町レジデンス 5 F(〒 102)
電話　03-238-1321(営業)
03-238-1323(編集)
印刷所　新村印刷株式会社

ISBN4-89369-012-4 C3055 ¥3900E

通常払込料金
加入者負担

払込通知票

各票の※印欄は、払込人において記載してください。

口座番号	東京	7	–	18397
加入者名	株式会社ビー・エヌ・エヌ			

金額	億	千	百	十	万	千	百	十	円
※									

払込み	特殊
	円

払込人住所氏名　※（郵便番号　　　　　）

備考

受付局日附印

記載事項を訂正した場合は、その箇所に訂正印を押してください。

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

ここから切り離して御使用ください。

通常払込料金
加入者負担

払込票

口座番号	東京	7	–	18397
加入者名	株式会社ビー・エヌ・エヌ			

金額	億	千	百	十	万	千	百	十	円
※									

払込人住所氏名

備考

受付局日附印

切り取らないで郵便局にお出しください。

右欄にお名前・御住所・御連絡先をお書き込みの上、お近くの郵便局へ料金を添えてお持ちください。

切り取らないで郵便局にお出しください。

この欄は、加入者あての通信にお使いください。

注：フリガナを忘れずお書き下さい。

注文書

お名前		電話	自宅
			勤務先
御住所	〒		

書籍／ディスク名	定価	数量	備考
	円		
	円		
	円		
	円		

＊郵送料は書籍各300円、ディスク各500円です。

＊ディスクご注文の際は必ずメディアを明記して下さい。

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

PC-9801シリーズ テクニカルノウハウ

PC-Techknow

BNN
Bug News Network

Writing DMSC

98V



PC-TECHKNOW 98V
Hardly a day goes by that technology doesn't touch our lives in some new way. Everything is getting faster, cheaper, and more confusing. And one of the most exciting things happening is the information and telecommunications age that is dawning all around us.

ISBN4-89369-012-4 C3055 ¥3900E

定価3,900円